

# P GVCRR EXHIBIT 1002

## PART 4

uP Data 0-7, uP Data P, uPAdd 0-2, CS

5        These signals are used by a microprocessor to address the programmable registers within the chip. The odd parity signal uP Data P is only checked when data is written to the Fifo Data or Checksum Registers and microprocessor parity is enabled.

Clk

10        The clock input is used to generate some of the chip timing. It is expected to be in the 10-20 Mhz range.

Read En, Write En

15        During microprocessor accesses, while CS is true, these signals determine the direction of the microprocessor accesses. During data transfers in the WD mode these signals are data strobes used in conjunction with Port A Ack.

Port B 00-07, 10-17, 20-27, 30-37, 0P-3P

20        Port B is a 32 bit data port. There is one odd parity bit for each byte. Port B 0P is the parity of bits 00-07, PortB 1P is the parity of bits 10-17, Port B 2P is the parity of bits 20-27, and Port B 3P is the parity of bits 30-37.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSE/7209.001

8/24/89-7

**B Select, B Req, B Ack, Parity Sync, B Output Enable**

These signals are used in the data transfer mode to control the handshake of data on Port B. Port B Req and Port B Ack are both gated with Port B Select.

5 The Port B Ack signal is used to strobe the data on the Port B data lines. The parity sync signal is used to indicate to a chip configured as the parity chip to indicate that the last words of data involved in the parity accumulation are on Port B. The Port B data  
10 lines will only be driven by the Fifo chip if all of the following conditions are met:

- a. the data transfer is from Port A to Port B;
- b. the Port B select signal is true;
- c. the Port B output enable signal is true; and
- 15 d. the chip is not configured as the parity chip or it is in parity correct mode and the Parity Sync signal is true.

**Reset**

This signal resets all the registers within the  
20 chip and causes all bidirectional pins to be in a high impedance state.

**DESCRIPTION OF OPERATION**

Normal Operation. Normally the chip acts as a simple FIFO chip. A FIFO is simulated by using two RAM  
25 buffers in a simple ping-pong mode. It is intended, but not mandatory, that data is burst into or out of

the FIFO on Port B. This is done by holding Port B Sel signal low and pulsing the Port B Ack signal. When transferring data from Port B to Port A, data is first written into RAM X and when this is full, the data paths will be switched such that Port B may start writing to RAM Y. Meanwhile the chip will begin emptying RAM X to Port A. When RAM Y is full and RAM X empty the data paths will be switched again such that Port B may reload RAM X and Port A may empty RAM Y.

5  
10       Port A Slave Mode. This is the default mode and the chip is reset to this condition. In this mode the chip waits for a master such as one of the SCSI adapter chips 542 to raise Port A Request for data transfer. If data is available the Fifo chip will respond with Port A Ack/Rdy.

15  
20       Port A WD Mode. The chip may be configured to run in the WD or Western Digital mode. In this mode the chip must be configured as a slave on Port A. It differs from the default slave mode in that the chip responds with Read Enable or Write Enable as appropriate together with Port A Ack/Rdy. This mode is intended to allow the chip to be interfaced to the Western Digital 33C93A SCSI chip or the NCR 53C90 SCSI chip.

25       Port A Master Mode. When the chip is configured as a master, it will raise Port A Ack/Rdy when it is

ready for data transfer. This signal is expected to be tied to the Request input of a DMA controller which will respond with Port A Req when data is available. In order to allow the DMA controller to burst, the Port A Ack/Rdy signal will only be negated after every 8 or 16 bytes transferred.

Port B Parallel Write Mode. In parallel write mode, the chip is configured to be the parity chip for a parallel transfer from Port B to Port A. In this mode, when Port B Select and Port B Request are asserted, data is written into RAM X or RAM Y each time the Port B Ack signal is received. For the first block of 128 bytes data is simply copied into the selected RAM. The next 128 bytes driven on Port B will be exclusive-ORed with the first 128 bytes. This procedure will be repeated for all drives such that the parity is accumulated in this chip. The Parity Sync signal should be asserted to the parallel chip together with the last block of 128 bytes. This enables the chip to switch access to the other RAM and start accumulating a new 128 bytes of parity.

Port B Parallel Read Mode - Check Data. This mode is set if all drives are being read and parity is to be checked. In this case the Parity Correct bit in the Data Transfer Configuration Register is not set. The parity chip will first read 128 bytes on Port A as

in a normal read mode and then raise Port B Request. While it has this signal asserted the chip will monitor the Port B Ack signals and exclusive-or the data on Port B with the data in its selected RAM. The Parity Sync should again be asserted with the last block of 128 bytes. In this mode the chip will not drive the Port B data lines but will check the output of its exclusive-or logic for zero. If any bits are set at this time a parallel parity error will be flagged.

5  
10       Port B Parallel Read Mode - Correct Data. This mode is set by setting the Parity Correct bit in the Data Transfer Configuration Register. In this case the chip will work exactly as in the check mode except that when Port B Output Enable, Port B Select and Parity Sync are true the data is driven onto the Port B data lines and a parallel parity check for zero is not performed.

15  
20       Byte Swap. In the normal mode it is expected that Port B bits 00-07 are the first byte, bits 10-17 the second byte, bits 20-27 the third byte, and bits 30-37 the last byte of each word. The order of these bytes may be changed by writing to the byte swap bits in the configuration register such that the byte address bits are inverted. The way the bytes are written and read also depend on whether the CPU interface is configured as 16 or 8 bits. The following

25

table shows the byte alignments for the different possibilities for data transfer using the Port A Request / Acknowledge handshake:

CPU I/F	Invert Addr 1	Invert Addr 0	Port B 00-07	Port B 10-17	Port B 20-27	Port B 30-37
5						
8	False	False	Port A byte 0	Port A byte 1	Port A byte 2	Port A byte 1
10						
8	False	True	Port A byte 1	Port A byte 0	Port A byte 3	Port A byte 2
8	True	False	Port A byte 2	Port A byte 3	Port A byte 0	Port A byte 1
8	True	True	Port A byte 3	Port A byte 2	Port A byte 1	Port A byte 0
15						
16	False	False	Port A byte 0	uProc byte 0	Port A byte 1	uProc byte 1
16	False	True	uProc byte 0	Port A byte 0	uProc byte 1	Port A byte 1
20						
16	True	False	Port A byte 1	uProc byte 1	Port A byte 0	uProc byte 0
16	True	True	uProc byte 1	Port A byte 1	uProc byte 0	Port A byte 0

When the Fifo is accessed by reading or writing the Fifo Data Register through the microprocessor port in 8 bit mode, the bytes are in the same order as the table above but the uProc data port is used instead of Port A. In 16 bit mode the table above applies.

Odd Length Transfers. If the data transfer is not a multiple of 32 words, or 128 bytes, the microprocessor must manipulate the internal registers of the chip to ensure all data is transferred. Port A Ack and Port B Req are normally not asserted until all

32 words of the selected RAM are available. These signals may be forced by writing to the appropriate RAM status bits of the Data Transfer Status Register.

5 When an odd length transfer has taken place the microprocessor must wait until both ports are quiescent before manipulating any registers. It should then reset both of the Enable Data Transfer bits for Port A and Port B in the Data Transfer Control Register. It must then determine by reading their Address Registers  
10 and the RAM Access Control Register whether RAM X or RAM Y holds the odd length data. It should then set the corresponding Address Register to a value of 20 hexadecimal, forcing the RAM full bit and setting the address to the first word. Finally the microprocessor  
15 should set the Enable Data Transfer bits to allow the chip to complete the transfer.

At this point the Fifo chip will think that there are now a full 128 bytes of data in the RAM and will transfer 128 bytes if allowed to do so. The fact that  
20 some of these 128 bytes are not valid must be recognized externally to the FIFO chip.



PROGRAMMABLE REGISTERS

Data Transfer Configuration Register (Read/Write)

Register Address 0. This register is cleared by the reset signal.

- 5           Bit 0    WD Mode. Set if data transfers are to use the Western Digital WD33C93A protocol, otherwise the Adaptec 6250 protocol will be used.
- 10           Bit 1    Parity Chip. Set if this chip is to accumulate Port B parities.
- Bit 2    Parity Correct Mode. Set if the parity chip is to correct parallel parity on Port B.
- 15           Bit 3    CPU Interface 16 bits wide. If set, the microprocessor data bits are combined with the Port A data bits to effectively produce a 16 bit Port. All accesses by the microprocessor as well as all data transferred using the Port A Request and Acknowledge handshake will transfer 16 bits.
- 20           Bit 4    Invert Port A byte address 0. Set to invert the least significant bit of Port A byte address.
- 25           Bit 5    Invert Port A byte address 1. Set to invert the most significant bit of Port A byte address.
- Bit 6    Checksum Carry Wrap. Set to enable the carry out of the 16 bit checksum adder to carry back into the least significant bit of the adder.
- 30           Bit 7    Reset. Writing a 1 to this bit will reset the other registers. This bit resets itself after a maximum of 2 clock cycles and will therefore normally be read as a 0. No other register should be written for a minimum of 4 clock cycles after writing to this bit.
- 35

Data Transfer Control Register (Read/Write)

Register Address 1. This register is cleared by the reset signal or by writing to the reset bit.

- 5 Bit 0 Enable Data Transfer on Port A. Set to enable the Port A Req/Ack handshake.
- 10 Bit 1 Enable Data Transfer on Port B. Set to enable the Port B Req/Ack handshake.
- 15 Bit 2 Port A to Port B. If set, data transfer is from Port A to Port B. If reset, data transfer is from Port B to Port A. In order to avoid any glitches on the request lines, the state of this bit should not be altered at the same time as the enable data transfer bits 0 or 1 above.
- 20 Bit 3 uProcessor Parity Enable. Set if parity is to be checked on the microprocessor interface. It will only be checked when writing to the Fifo Data Register or reading from the Fifo Data or Checksum Registers, or during a Port A Request/Acknowledge transfer in 16 bit mode. The chip will, however, always re-generate parity ensuring that correct parity is written to the RAM or read on the microprocessor interface.
- 25 Bit 4 Port A Parity Enable. Set if parity is to be checked on Port A. It is checked when accessing the Fifo Data Register in 16 bit mode, or during a Port A Request/Acknowledge transfer. The chip will, however, always re-generate parity ensuring that correct parity is written to the RAM or read on the Port A interface.
- 30 Bit 5 Port B Parity Enable. Set if Port B data has valid byte parities. If it is not set, byte parity is generated internally to the chip when writing to the RAMs. Byte parity is not checked when writing from Port B, but always checked when reading to Port B.
- 35
- 40

- 5           Bit 6    Checksum Enable.   Set to enable writing to the 16 bit checksum register. This register accumulates a 16 bit checksum for all RAM accesses, including accesses to the Fifo Data Register, as well as all writes to the checksum register. This bit must be reset before reading from the Checksum Register.
- 10           Bit 7    Port A Master.   Set if Port A is to operate in the master mode on Port A during the data transfer.

Data Transfer Status Register (Read Only)

Register Address 2. This register is cleared by the reset signal or by writing to the reset bit.

- 15           Bit 0    Data in RAM X or RAM Y. Set if any bits are true in the RAM X, RAM Y, or Port A byte address registers.
- 20           Bit 1    uProc Port Parity Error. Set if the uProc Parity Enable bit is set and a parity error is detected on the microprocessor interface during any RAM access or write to the Checksum Register in 16 bit mode.
- 25           Bit 2    Port A Parity Error. Set if the Port A Parity Enable bit is set and a parity error is detected on the Port A interface during any RAM access or write to the Checksum Register.
- 30           Bit 3    Port B Parallel Parity Error. Set if the chip is configured as the parity chip, is not in parity correct mode, and a non zero result is detected when the Parity Sync signal is true. It is also set whenever data is read out onto Port B and the data being read back through the bidirectional buffer does not compare.
- 35
- 40           Bits 4-7 Port B Bytes 0-3 Parity Error. Set whenever the data being read out of the RAMs on the Port B side has bad parity.

Ram Access Control Register (Read/Write)

Register Address 3. This register is cleared by the reset signal or by writing to the reset bit. The Enable Data Transfer bits in the Data Transfer Control Register must be reset before attempting to write to this register, else the write will be ignored.

- 5 Bit 0 Port A byte address 0. This bit is the least significant byte address bit. It is read directly bypassing any inversion done by the invert bit in the Data Transfer Configuration Register.
- 10 Bit 1 Port A byte address 1. This bit is the most significant byte address bit. It is read directly bypassing any inversion done by the invert bit in the Data Transfer Configuration Register.
- 15 Bit 2 Port A to RAM Y. Set if Port A is accessing RAM Y, and reset if it is accessing RAM X .
- 20 Bit 3 Port B to RAM Y. Set if Port B is accessing RAM Y, and reset if it is accessing RAM X .
- 25 Bit 4 Long Burst. If the chip is configured to transfer data on Port A as a master, and this bit is reset, the chip will only negate Port A Ack/Rdy after every 8 bytes, or 4 words in 16 bit mode, have been transferred. If this bit is set, Port A Ack/Rdy will be negated every 16 bytes, or 8 words in 16 bit mode.
- 30 Bits 5-7 Not Used.

RAM X Address Register (Read/Write)

Register Address 4. This register is cleared by the reset signal or by writing to the reset bit. The Enable Data Transfer bits in the Data Transfer Control

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Register must be reset before attempting to write to this register, else the write will be ignored.

Bits 0-4           RAM X word address  
Bit   5            RAM X full  
5       Bits 6-7       Not Used

RAM Y Address Register (Read/Write)

Register Address 5. This register is cleared by the reset signal or by writing to the reset bit. The Enable Data Transfer bits in the Data Transfer Control Register must be reset before attempting to write to this register, else the write will be ignored.

10           Bits 0-4           RAM Y word address  
              Bit   5            RAM Y full  
              Bits 6-7        Not Used

15           Fifo Data Register (Read/Write)

Register Address 6. The Enable Data Transfer bits in the Data Transfer Control Register must be reset before attempting to write to this register, else the write will be ignored. The Port A to Port B bit in the Data Transfer Control register must also be set before writing this register. If it is not, the RAM controls will be incremented but no data will be written to the RAM. For consistency, the Port A to Port B should be reset prior to reading this register.

Bits 0-7 are Fifo Data. The microprocessor may access the FIFO by reading or writing this register. The RAM control registers are updated as if the access was using Port A. If the chip is configured with a 16 bit CPU Interface the most significant byte will use the Port A 0-7 data lines, and each Port A access will increment the Port A byte address by 2.

Port A Checksum Register (Read/Write)

Register Address 7. This register is cleared by the reset signal or by writing to the reset bit.

Bits 0-7 are Checksum Data. The chip will accumulate a 16 bit checksum for all Port A accesses. If the chip is configured with a 16 bit CPU interface, the most significant byte is read on the Port A 0-7 data lines. If data is written directly to this register it is added to the current contents rather than overwriting them. It is important to note that the Checksum Enable bit in the Data Transfer Control Register must be set to write this register and reset to read it.

PROGRAMMING THE FIFO CHIP

In general the fifo chip is programmed by writing to the data transfer configuration and control registers to enable a data transfer, and by reading

the data transfer status register at the end of the transfer to check the completion status. Usually the data transfer itself will take place with both the Port A and the Port B handshakes enabled, and in this case  
5 the data transfer itself should be done without any other microprocessor interaction. In some applications, however, the Port A handshake may not be enabled, and it will be necessary for the microprocessor to fill or empty the fifo by repeatedly  
10 writing or reading the Fifo Data Register.

Since the fifo chip has no knowledge of any byte counts, there is no way of telling when any data transfer is complete by reading any register within this chip itself. Determination of whether the data  
15 transfer has been completed must therefore be done by some other circuitry outside this chip.

The following C language routines illustrate how the parity FIFO chip may be programmed. The routines assume that both Port A and the microprocessor port are  
20 connected to the system microprocessor, and return a size code of 16 bits, but that the hardware addresses the Fifo chip as long 32 bit registers.

```
struct FIFO_regs {
    unsigned char config,a1,a2,a3 ;
    unsigned char control,b1,b2,b3;
    unsigned char status,c1,c2,c3;
5   unsigned char ram_access_control,d1,d2,d3;
    unsigned char ram_X_addr,e1,e2,e3;
    unsigned char ram_Y_addr,f1,f2,f3;
    unsigned long data;
    unsigned int checksum,h1;
10  };

#define FIFO1 ((struct FIFO_regs*) FIFO_BASE_ADDRESS)

#define FIFO_RESET 0x80
#define FIFO_16_BITS 0x08
#define FIFO_CARRY_WRAP 0x40
15  #define FIFO_PORT_A_ENABLE 0x01
    #define FIFO_PORT_B_ENABLE 0x02
    #define FIFO_PORT_ENABLES 0x03
    #define FIFO_PORT_A_TO_B 0x04
    #define FIFO_CHECKSUM_ENABLE 0x40
20  #define FIFO_DATA_IN_RAM 0x01
    #define FIFO_FORCE_RAM_FULL 0x20

#define PORT_A_TO_PORT_B(fifo) ((fifo-> control ) & 0x04)
#define PORT_A_BYTE_ADDRESS(fifo) ((fifo->ram_access_control) &
    0x03)
25  #define PORT_A_TO_RAM_Y(fifo) ((fifo->ram_access_control ) &
    0x04)
    #define PORT_B_TO_RAM_Y(fifo) ((fifo-> ram_access_control ) &
    0x08)

/*****
30  The following routine initiates a Fifo data transfer using
two values passed to it.

    config_data    This is the data to be written to the
                    configuration register.

    control_data   This is the data to be written to the Data
35  Transfer Control Register.  If the data transfer
is to take place automatically using both the
Port A and Port B handshakes, both data transfer
enables bits should be set in this parameter.
*****/

40  FIFO_initiate_data_transfer(config_data, control_data)
unsigned char config_data, control_data;
{
    FIFO1->config = config_data | FIFO_RESET;    /* Set
        Configuration value & Reset */
}
```



```
FIFO1->control = control_data & (~FIFO_PORT_ENABLES); /* Set
everything but enables */
FIFO1->control = control_data ; /* Set data transfer
enables */
5 }

/*****
The following routine forces the transfer of any odd bytes
that have been left in the Fifo at the end of a data transfer.
It first disables both ports, then forces the Ram Full bits, and
10 then re-enables the appropriate Port.
*****/

FIFO_force_odd_length_transfer()
{
FIFO1->control &= ~FIFO_PORT_ENABLES; /* Disable Ports A & B */
15 if (PORT_A_TO_PORT_B(FIFO1)) {
if (PORT_A_TO_RAM_Y(FIFO1)) {
FIFO1->ram_Y_addr = FIFO_FORCE_RAM_FULL; /* Set RAM Y
full */
}
20 else FIFO1->ram_X_addr = FIFO_FORCE_RAM_FULL ; /* Set RAM
X full */
FIFO1->control |= FIFO_PORT_B_ENABLE ; /* Re-Enable
Port B */
}
25 else {
if (PORT_B_TO_RAM_Y(FIFO1)) {
FIFO1->ram_Y_addr = FIFO_FORCE_RAM_FULL ; /* Set
RAM Y full */
}
30 else FIFO1->ram_X_addr = FIFO_FORCE_RAM_FULL ; /* Set RAM
X full */
FIFO1->control |= FIFO_PORT_A_ENABLE ; /* Re-Enable
Port A */
}
35 }

/*****
The following routine returns how many odd bytes have been
left in the Fifo at the end of a data transfer.
*****/

40 int FIFO_count_odd_bytes()
{
int number_odd_bytes;
number_odd_bytes=0;
if (FIFO1->status & FIFO_DATA_IN_RAM) {
45 if (PORT_A_TO_PORT_B(FIFO1)) {
number_odd_bytes = (PORT_A_BYTE_ADDRESS(FIFO1)) ;
if (PORT_A_TO_RAM_Y(FIFO1))
number_odd_bytes += (FIFO1->ram_Y_addr) * 4 ;
}
```

```

    else number_odd_bytes += (FIFO1->ram_X_addr) * 4 ;
  }
  else {
    if (PORT_B_TO_RAM_Y(FIFO1))
      number_odd_bytes = (FIFO1->ram_Y_addr) * 4 ;
    else number_odd_bytes = (FIFO1->ram_X_addr) * 4 ;
  }
}
return (number_odd_bytes);
}

/*****
The following routine tests the microprocessor interface of
the chip. It first writes and reads the first 6 registers. It
then writes 1s, 0s, and an address pattern to the RAM, reading the
data back and checking it.

The test returns a bit significant error code where each
bit represents the address of the registers that failed.

Bit 0 = config register failed
Bit 1 = control register failed
Bit 2 = status register failed
Bit 3 = ram access control register failed
Bit 4 = ram X address register failed
Bit 5 = ram Y address register failed
Bit 6 = data register failed
Bit 7 = checksum register failed
*****/

#define RAM_DEPTH 64 /* number of long words in Fifo Ram */
reg_expected_data[6] = { 0x7F, 0xFF, 0x00, 0x1F, 0x3F, 0x3F };

char FIFO_uprocessor_interface_test()
{
  unsigned long test_data;
  char *register_addr;
  int i;
  char j,error;
  FIFO1->config = FIFO_RESET; /* reset the chip */
  error=0;
  register_addr =(char *) FIFO1;
  j=1;

  /* first test registers 0 thru 5 */

  for (i=0; i<6; i++) {
    *register_addr = 0xFF; /* write test data */
    if (*register_addr != reg_expected_data[i]) error |= j;
    *register_addr = 0; /* write 0s to register */
    if (*register_addr) error |= j;
  }
}

```

```

5
    *register_addr = 0xFF;          /* write test data again */
    if (*register_addr != reg_expected_data[i]) error != j;
    FIFO1->config = FIFO_RESET;     /* reset the chip */
    if (*register_addr) error != j; /* register should be 0 */
    register_addr++;                /* go to next register */
    j <<= 1;
}

/* now test Ram data & checksum registers
test 1s throughout Ram & then test 0s */

    for (test_data = -1; test_data != 1; test_data++) { /* test
for 1s & 0s */
        FIFO1->config = FIFO_RESET | FIFO_16_BITS ;
        FIFO1->control = FIFO_PORT_A_TO_B;
15        for (i=0; i<RAM_DEPTH; i++) /* write data to RAM */
            FIFO1->data = test_data;
        FIFO1->control = 0;
        for (i=0; i<RAM_DEPTH; i++)
20            if (FIFO1->data != test_data) error != j; /* read
& check data */
            if (FIFO1->checksum) error != 0x80; /* checksum
should = 0 */
        }
25

        /* now test Ram data with address pattern
        uses a different pattern for every byte */

        test_data=0x00010203; /* address pattern start */
        FIFO1->config = FIFO_RESET | FIFO_16_BITS | FIFO_CARRY_WRAP;
        FIFO1->control = FIFO_PORT_A_TO_B | FIFO_CHECKSUM_ENABLE;
30        for (i=0; i<RAM_DEPTH; i++) {
            FIFO1->data = test_data; /* write address pattern */
            test_data += 0x04040404;
        }
        test_data=0x00010203; /* address pattern start */
        FIFO1->control = FIFO_CHECKSUM_ENABLE;
        for (i=0; i<RAM_DEPTH; i++) {
            if (FIFO1->status != FIFO_DATA_IN_RAM)
                error != 0x04; /* should be data in ram */
40            if (FIFO1->data != test_data) error != j; /* read &
check address pattern */
            test_data += 0x04040404;
        }
        if (FIFO1->checksum != 0x0102) error != 0x80; /* test
45 checksum of address pattern */
        FIFO1->config = FIFO_RESET | FIFO_16_BITS ; /* inhibit carry
wrap */
        FIFO1->checksum = 0xFEFE; /* writing adds to checksum */

```

-132-

```
if (FIFO1->checksum) error !=0x80; /* checksum should be 0 */  
if (FIFO1->status) error != 0x04; /* status should be 0 */  
return (error);  
}
```

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

## CLAIMS

1. Network server apparatus for use with a data network and a mass storage device, comprising:

an interface processor unit coupleable to said network and to said mass storage device;

a host processor unit capable of running remote procedures defined by a client node on said network;

means in said interface processor unit for satisfying requests from said network to store data from said network on said mass storage device;

means in said interface processor unit for satisfying requests from said network to retrieve data from said mass storage device to said network; and

means in said interface processor unit for transmitting predefined categories of messages from said network to said host processor unit for processing in said host processor unit, said transmitted messages including all requests by a network client to run client-defined procedures on said network server apparatus.

2. Apparatus according to claim 1, wherein said interface processor unit comprises:

a network control unit coupleable to said network;

a data control unit coupleable to said mass storage device;

a buffer memory;

means in said network control unit for transmitting to said data control unit requests from said network to store specified storage data from said network on said mass storage device;

means in said network control unit for transmitting said specified storage data from said network to said buffer memory and from said buffer memory to said data control unit;

means in said network control unit for transmitting to said data control unit requests from said network to retrieve specified retrieval data from said mass storage device to said network;

means in said network control unit for transmitting said specified retrieval data from said data control unit to said buffer memory and from said buffer memory to said network; and

means in said network control unit for transmitting said predefined categories of messages from said network to said host processing unit for processing by said host processing unit.

3. Apparatus according to claim 2, wherein said data control unit comprises:

a storage processor unit coupleable to said mass storage device;

a file processor unit;

means on said file processor unit; for translating said file system level storage requests from said

network into requests to store data at specified physical storage locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to write data from said buffer memory into said specified physical storage locations in said mass storage device;

means on said file processor unit for translating file system level retrieval requests from said network into requests to retrieve data from specified physical retrieval locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to retrieve data from said specified physical retrieval locations in said mass storage device to said buffer memory if said data from said specified physical locations is not already in said buffer memory; and

means in said storage processor unit for transmitting data between said buffer memory and said mass storage device.

4. Network server apparatus for use with a data network and a mass storage device, comprising:

a network control unit coupleable to said network;

a data control unit coupleable to said mass storage device;

a buffer memory;

means for transmitting from said network control unit to said data control unit requests from said

network to store specified storage data from said network on said mass storage device;

means for transmitting said specified storage data by DMA from said network control unit to said buffer memory and by DMA from said buffer memory to said data control unit;

means for transmitting from said network control unit to said data control unit requests from said network to retrieve specified retrieval data from said mass storage device to said network; and

means for transmitting said specified retrieval data by DMA from said data control unit to said buffer memory and by DMA from said buffer memory to said network control unit.

5. Apparatus according to claim 1, for use further with a buffer memory, and wherein said requests from said network to store and retrieve data include file system level storage and retrieval requests respectively, and wherein said interface processor unit comprises:

a storage processor unit coupleable to said mass storage device;

a file processor unit;

means on said file processor unit for translating said file system level storage requests into requests to store data at specified physical storage locations in said mass storage device;



095447/4

means on said file processor unit for instructing said storage processor unit to write data from said buffer memory into said specified physical storage locations in said mass storage device;

means on said file processor unit for translating said file system level retrievable requests into requests to retrieve data from specified physical retrievable locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to retrieve data from said specified physical retrievable locations in said mass storage device to said buffer memory if said data from said specified physical locations is not already in said buffer memory; and

means in said storage processor unit for transmitting data between said buffer memory and said mass storage device.

6. Network server apparatus for use with a data network, comprising:

a network controller coupleable to said network to receive incoming information packets over said network, said incoming information packets including certain packets which contain part or all of a request to said server apparatus, said request being in either a first or a second class of requests to said server apparatus;

a first additional processor;

an interchange bus different from said network and coupled between said network controller and said first additional processor;

means in said network controller for detecting and satisfying requests in said first class of requests contained in said certain incoming information packets, said network controller lacking means in said network controller for satisfying requests in said second class of requests;

means in said network controller for detecting and assembling into assembled requests, requests in said second class of requests contained in said certain incoming information packets;

means for delivering said assembled requests from said network controller to said first additional processor over said interchange bus; and

means in said first additional processor for further processing said assembled requests in said second class of requests.

7. Apparatus according to claim 6 wherein said packets each include a network node destination address, and wherein said means in said network controller for detecting and assembling into assembled requests, assembles said assembled requests in a format which omits said network node destination addresses.

8. Apparatus according to claim 6 wherein said means in said network controller for detecting and satisfying requests in said first class of requests, assembles said requests in said first class of requests into assembled requests before satisfying said requests in said first class of requests.

9. Apparatus according to claim 6, wherein said packets each include a network node destination address, wherein said means in said network controller for detecting and assembling into assembled requests, assembles said assembled requests in a format which omits said network node destination addresses, and wherein said means in said network controller for detecting and satisfying requests in said first class of requests, assembles said requests in said first class of requests, in a format which omits said network node destination addresses, before satisfying said requests in said first class of requests.

10. Apparatus according to claim 6, wherein said means in said network controller for detecting and satisfying requests in said first class includes means for preparing an outgoing message in response to one of said first class of requests, means for packaging said outgoing message in outgoing information packets suitable for transmission over said network, and means for transmitting said outgoing information packets over said network.

11. Apparatus according to claim 6, further comprising a buffer memory coupled to said interchange bus, and wherein said means for delivering said assembled requests comprises:

means for transferring the contents of said assembled requests over said interchange bus into said buffer memory; and

means for notifying said first additional processor of the presence of said contents in said buffer memory.

12. Apparatus according to claim 6, wherein said means in said first additional processor for further processing said assembled requests includes means for preparing an outgoing message in response to one of said second class of requests, said apparatus further comprising means for delivering said outgoing message from said first additional processor to said network controller over said interchange bus, said network controller further comprising means in said network controller for packaging said outgoing message in outgoing information packets suitable for transmission over said network, and means in said network controller for transmitting said outgoing information packages over said network.

13. Apparatus according to claim 6, wherein said first class of requests comprises requests for an address of said server apparatus, and wherein said means in said network controller for detecting and satisfying requests in said first class comprises means for preparing a response packet to such an address request and means for transmitting said response packet over said network.

14. Apparatus according to claim 6 for use further with a second data network, said network controller being coupleable further to said second network, wherein said first class of requests comprises requests to route a message to a destination reachable over said second network, and wherein said means in said network controller for detecting and satisfying requests in said first class comprises means for detecting that one of said certain packets comprises a request to route a message contained in said one of said certain packets to a destination reachable over said second network, and means for transmitting said message over said second network.

15. Apparatus according to claim 14 for use further with a third data network, said network controller further comprising means in said network controller for detecting particular requests in said incoming information packets to route a message contained in said particular requests, to a destination reachable over said third network, said apparatus further comprising:

a second network controller coupled to said interchange bus and coupleable to said third data network;

means for delivering said message contained in said particular requests to said second network controller over said interchange bus; and

means in said second network controller for transmitting said message contained in said particular requests over said third network.

16. Apparatus according to claim 6, for use further with a third data network, said network controller further comprising means in said network controller for detecting particular requests in said incoming information packets to route a message contained in said particular requests, to a destination reachable over said third network, said apparatus further comprising:

a second network controller coupled to said interchange bus and coupleable to said third data network;

means for delivering said message contained in said particular requests to said second network controller over said interchange bus; and

means in said second network controller for transmitting said message contained in said particular requests over said third network.

17. Apparatus according to claim 6 for use further with a mass storage device, wherein said first additional processor comprises a data control unit coupleable to said mass storage device, wherein said second class of requests comprises remote calls to procedures for managing a file system in said mass storage device, and wherein said means in said first additional processor for further processing said assembled requests in said second class of requests comprises means for executing file system procedures on said mass storage device in response to said assembled requests.

18. Apparatus according to claim 17 wherein said file system procedures include a read procedure for reading data from said mass storage device,

said means in said first additional processor for further processing said assembled requests including means for reading data from a specified location in said mass storage device in response to a remote call to said read procedure,

said apparatus further including means for delivering said data to said network controller,

said network controller further comprising means on said network controller for packaging said data in outgoing information packets suitable for transmission over said network, and means for transmitting said outgoing information packets over said network.

19. Apparatus according to claim 18, wherein said means for delivering comprises:

a system buffer memory coupled to said interchange bus;

means in said data control unit for transferring said data over said interchange bus into said buffer memory; and

means in said network controller for transferring said data over said interchange bus from said system buffer memory to said network controller.

20. Apparatus according to claim 17, wherein said file system procedures include a read procedure for reading a specified number of bytes of data from said mass storage device beginning at an address specified in logical terms including a file system ID and a file ID, said means for executing file system procedures comprising:

means for converting the logical address specified in a remote call to said read procedure to a physical address; and

means for reading data from said physical address in said mass storage device.

21. Apparatus according to claim 20, wherein said mass storage device comprises a disk drive having a numbered tracks and sectors, wherein said logical address specifies said file system ID, said file ID,



and a byte offset, and wherein said physical address specifies a corresponding track and sector number.

22. Apparatus according to claim 17, wherein said file system procedures include a read procedure for reading a specified number of bytes of data from said mass storage device beginning at an address specified in logical terms including a file system ID and a file ID,

said data control unit comprising a file processor coupled to said interchange bus and a storage processor coupled to said interchange bus and coupleable to said mass storage device,

said file processor comprising means for converting the logical address specified in a remote call to said read procedure to a physical address,

said apparatus further comprising means for delivering said physical address to said storage processor,

said storage processor comprising means for reading data from said physical address in said mass storage device and for transferring said data over said interchange bus into said buffer memory; and

means in said network controller for transferring said data over said interchange bus from said system buffer memory to said network controller.

23. Apparatus according to claim 17, wherein said file system procedures include a write procedure for

writing data contained in an assembled request, to said mass storage device,

said means in said first additional processor for further processing said assembled requests including means for writing said data to a specified location in said mass storage device in response to a remote call to said read procedure.

24. Apparatus according to claim 6, wherein said first additional processor comprises a host computer coupled to said interchange bus, wherein said second class of requests comprises remote calls to procedures other than procedures for managing a file system, and wherein said means in said first additional processor for further processing said assembled requests in said second class of requests comprises means for executing remote procedure calls in response to said assembled requests.

25. Apparatus according to claim 24, for use further with a mass storage device and a data control unit coupleable to said mass storage device and coupled to said interchange bus, wherein said network controller further comprises means in said network controller for detecting and assembling remote calls, received over said network, to procedures for managing a file system in said mass storage device, and wherein said data control unit comprises means for executing file system procedures on said mass storage device in

response to said remote calls to procedures for managing a file system in said mass storage device.

26. Apparatus according to claim 24, further comprising means for delivering all of said incoming information packets not recognized by said network controller to said host computer over said interchange bus.

27. Apparatus according to claim 26, wherein said network controller comprises:

a microprocessor;

a local instruction memory containing local instruction code;

a local bus coupled between said microprocessor and said local instruction memory;

bus interface means for interfacing said microprocessor with said interchange bus at times determined by said microprocessor in response to said local instruction code; and

network interface means for interfacing said microprocessor with said data network,

said local instruction memory including all instruction code necessary for said microprocessor to perform said function of detecting and satisfying requests in said first class of requests, and all instruction code necessary for said microprocessor to perform said function of detecting and assembling into

assembled requests, requests in said second class of requests.

28. Network server apparatus for use with a data network, comprising:

a network controller coupleable to said network to receive incoming information packets over said network, said incoming information packets including certain packets which contain part or all of a message to said server apparatus, said message being in either a first or a second class of messages to said server apparatus, said messages in said first class of messages including certain messages containing requests;

a host computer;

an interchange bus different from said network and coupled between said network controller and said host computer;

means in said network controller for detecting and satisfying said requests in said first class of messages ;

means for delivering messages in said second class of messages from said network controller to said host computer over said interchange bus; and

means in said host computer for further processing said messages in said second class of messages.

29. Apparatus according to claim 28, wherein said packets each include a network node destination address, and wherein said means for delivering messages

in said second class of messages comprises means in said network controller for detecting said messages in said second class of messages and assembling them into assembled messages in a format which omits said network node destination addresses.

30. Apparatus according to claim 28, wherein said means in said network controller for detecting and satisfying requests in said first class includes means for preparing an outgoing message in response to one of said requests in said first class of messages, means for packaging said outgoing message in outgoing information packets suitable for transmission over said network, and means for transmitting said outgoing information packets over said network.

31. Apparatus according to claim 28, for use further with a second data network, said network controller being coupleable further to said second network, wherein said first class of messages comprises messages to be routed to a destination reachable over said second network, and wherein said means in said network controller for detecting and satisfying requests in said first class comprises means for detecting that one of said certain packets includes a request to route a message contained in said one of said certain packets to a destination reachable over said second network, and means for transmitting said message over said second network.

32. Apparatus according to claim 28, for use further with a third data network, said network controller further comprising means in said network controller for detecting particular messages in said incoming information packets to be routed to a destination reachable over said third network, said apparatus further comprising:

a second network controller coupled to said interchange bus and coupleable to said third data network;

means for delivering said particular messages to said second network controller over said interchange bus, substantially without involving said host computer; and

means in said second network controller for transmitting said message contained in said particular requests over said third network, substantially without involving said host computer.

33. Apparatus according to claim 28, for use further with a mass storage device, further comprising a data control unit coupleable to said mass storage device,

said network controller further comprising means in said network controller for detecting ones of said incoming information packets containing remote calls to procedures for managing a file system in said mass storage device, and means in said network controller

095447/2

for assembling said remote calls from said incoming packets into assembled calls, substantially without involving said host computer,

said apparatus further comprising means for delivering said assembled file system calls to said data control unit over said interchange bus substantially without involving said host computer, and said data control unit comprising means in said data control unit for executing file system procedures on said mass storage device in response to said assembled file system calls, substantially without involving said host computer.

34. Apparatus according to claim 28, further comprising means for delivering all of said incoming information packets not recognized by said network controller to said host computer over said interchange bus.

35. Apparatus according to claim 28, wherein said network controller comprises:

a microprocessor;

a local instruction memory containing local instruction code;

a local bus coupled between said microprocessor and said local instruction memory;

bus interface means for interfacing said microprocessor with said interchange bus at times

- 151 -

Attorney Docket No.: AUSP7209MCF/GBR/WSW  
wsu/ausp/7209.claims

095447/2

determined by said microprocessor in response to said local instruction code; and

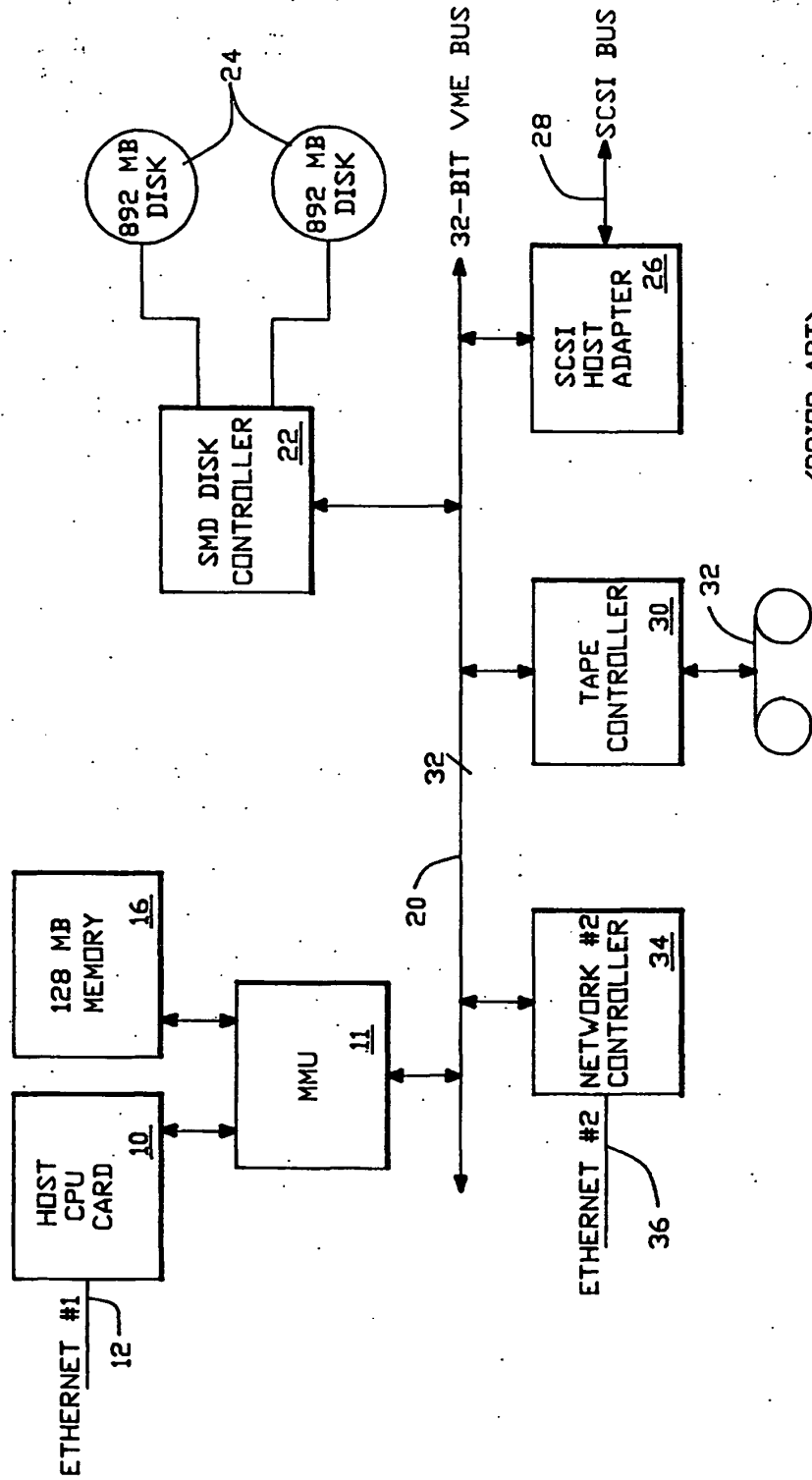
network interface means for interfacing said microprocessor with said data network,

said local instruction memory including all instruction code necessary for said microprocessor to perform said function of detecting and satisfying requests in said first class of requests.

For the Applicant,

  
Sanford T. Colb & Co.  
C:11071





(PRIOR ART)

FIG.-1

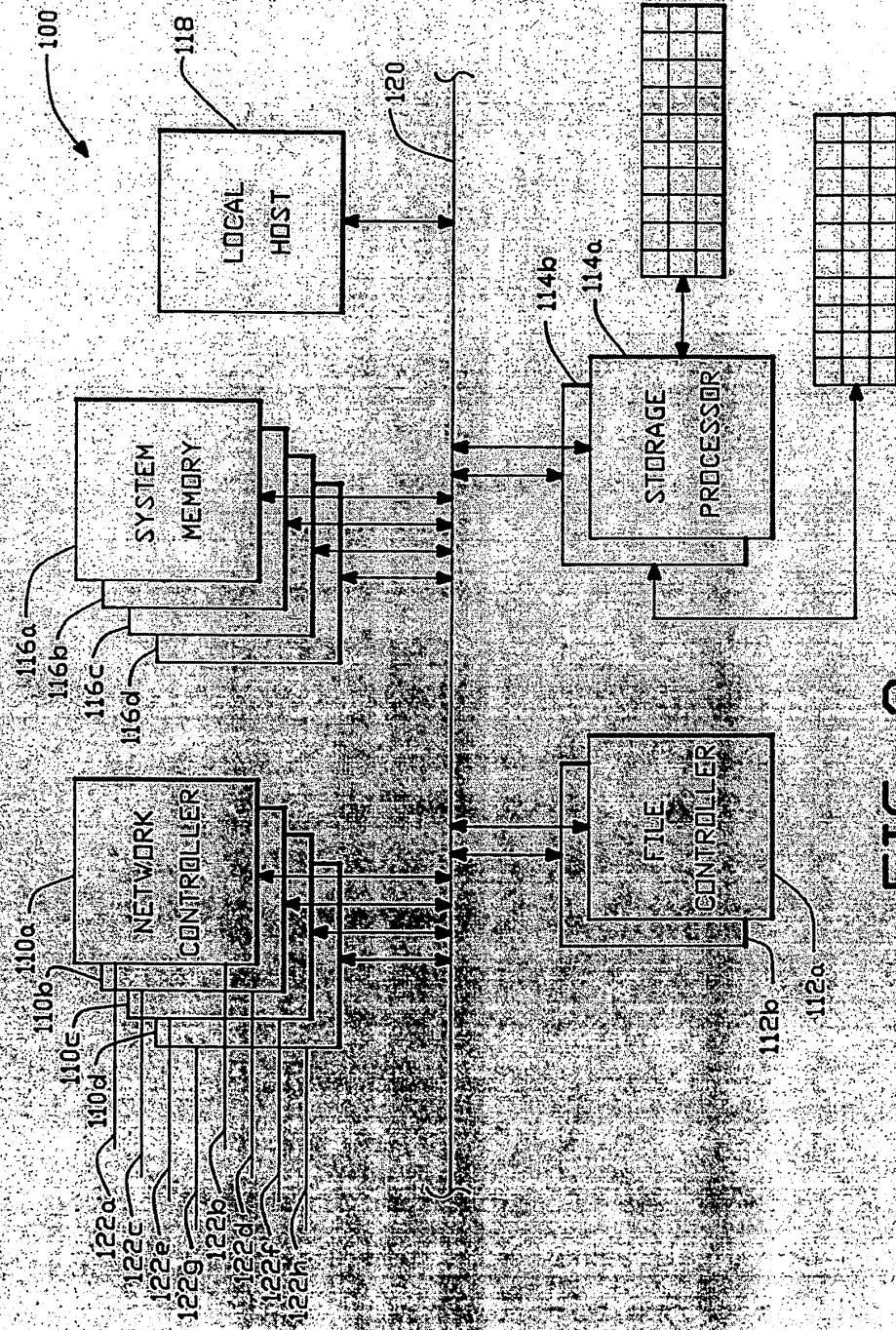


FIG.-2

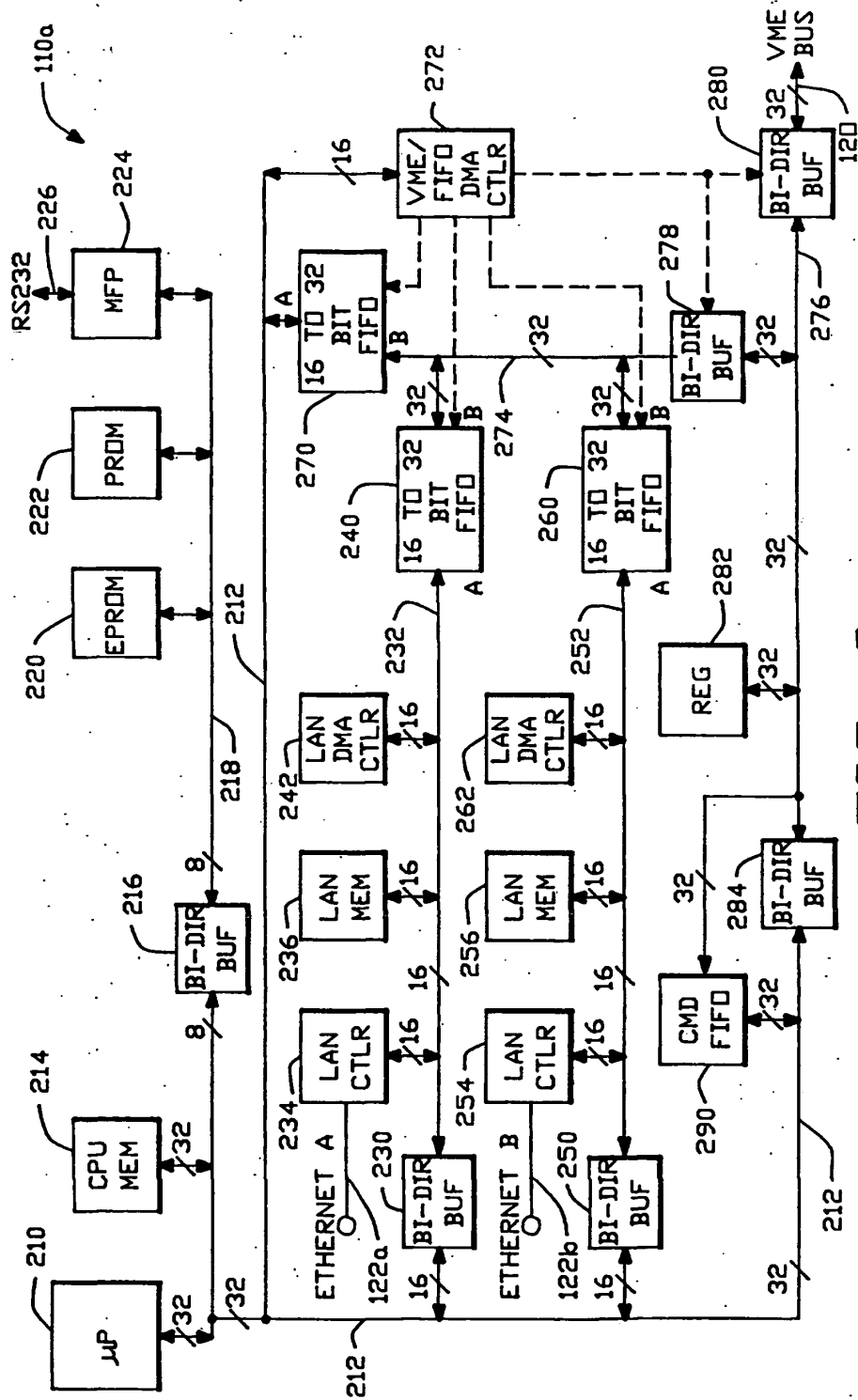
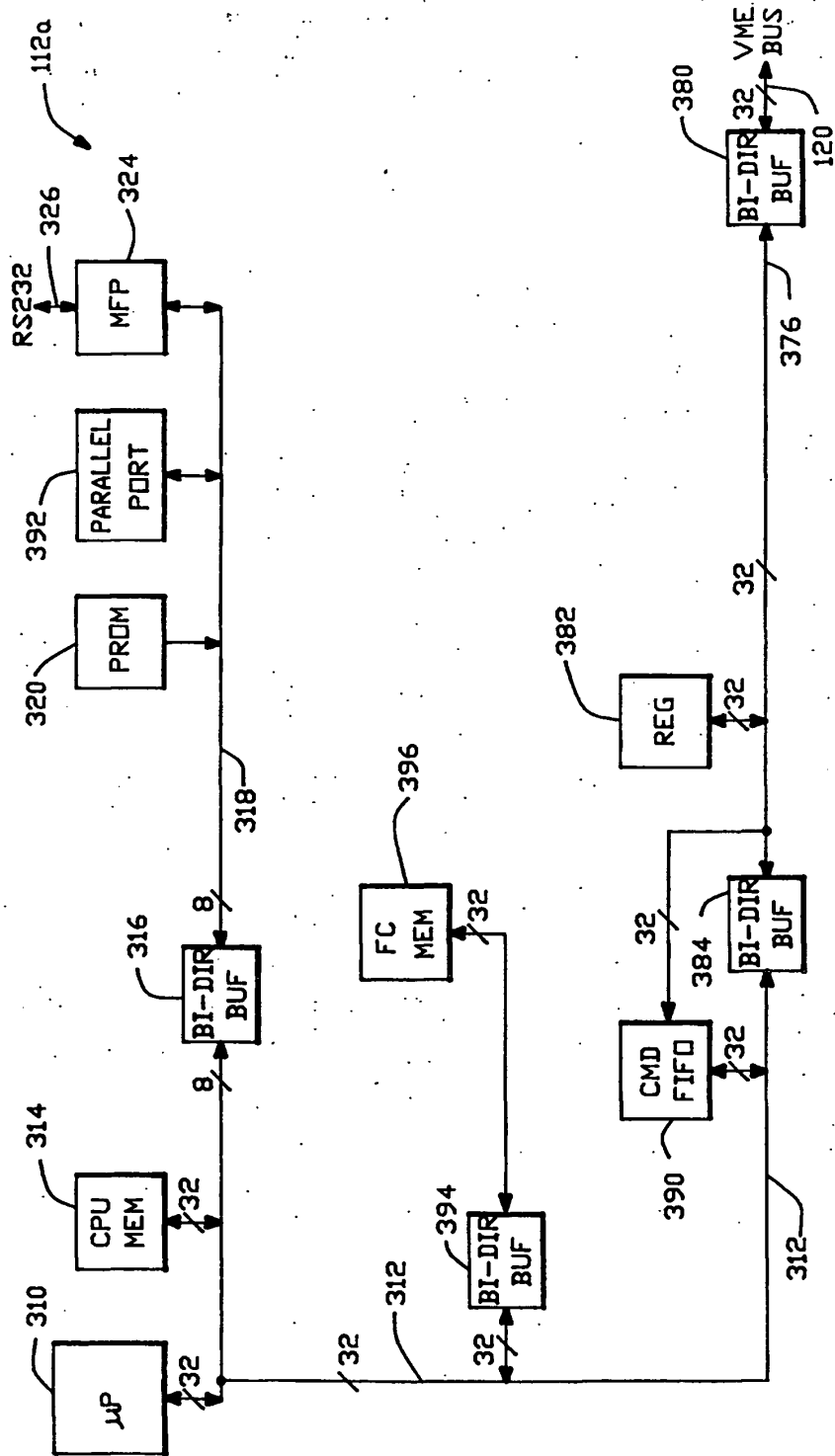


FIG.-3 (NETWORK CONTROLLER)



(FILE CONTROLLER)

FIG.-4

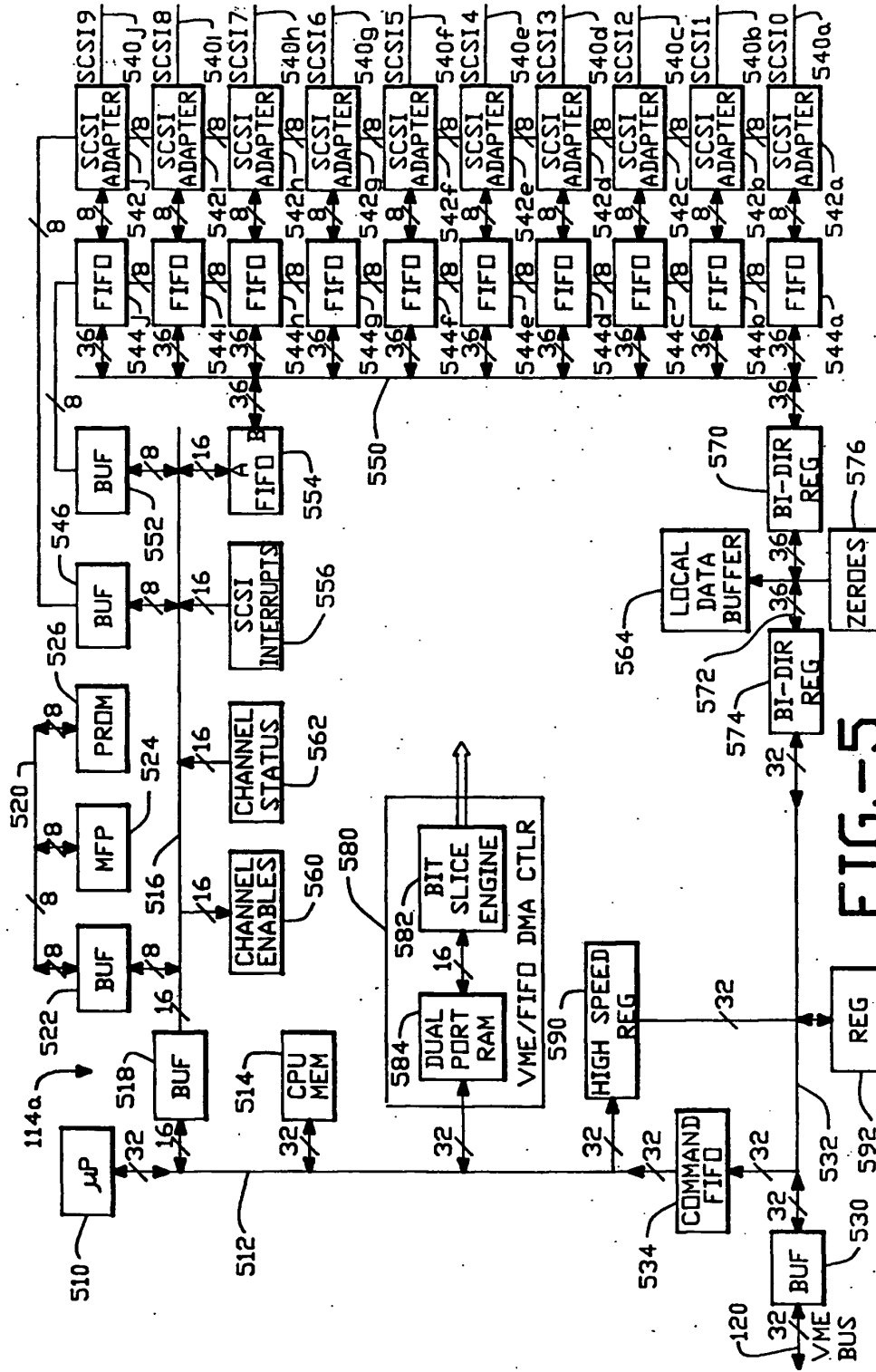


FIG. 5

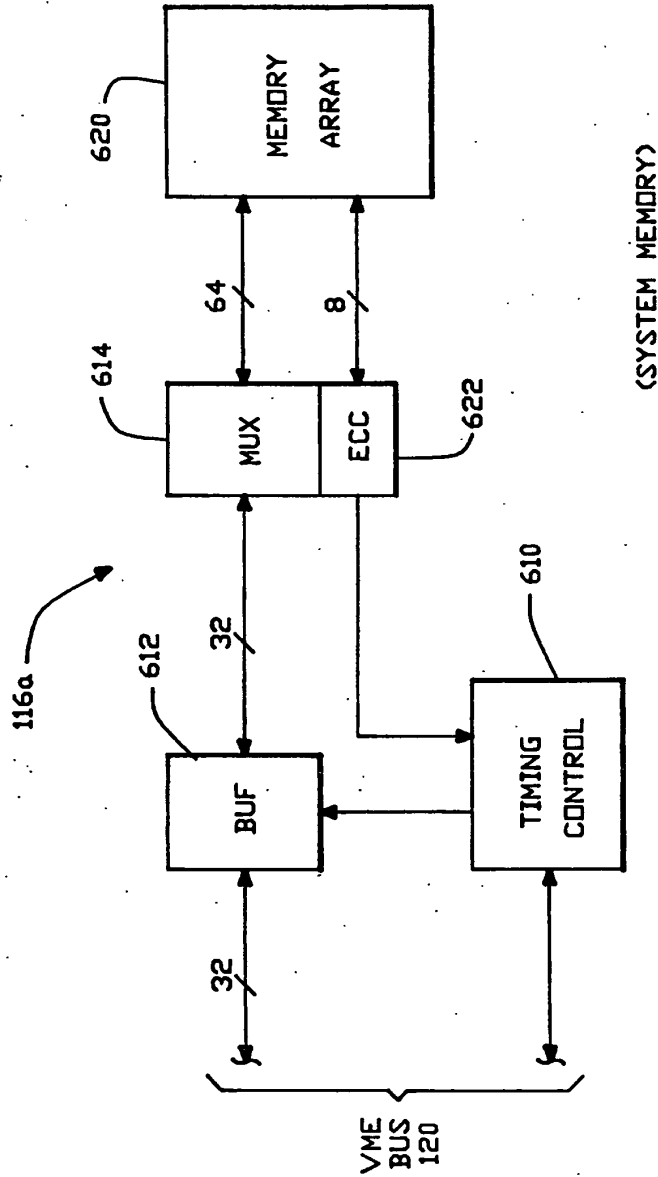


FIG.-6

12

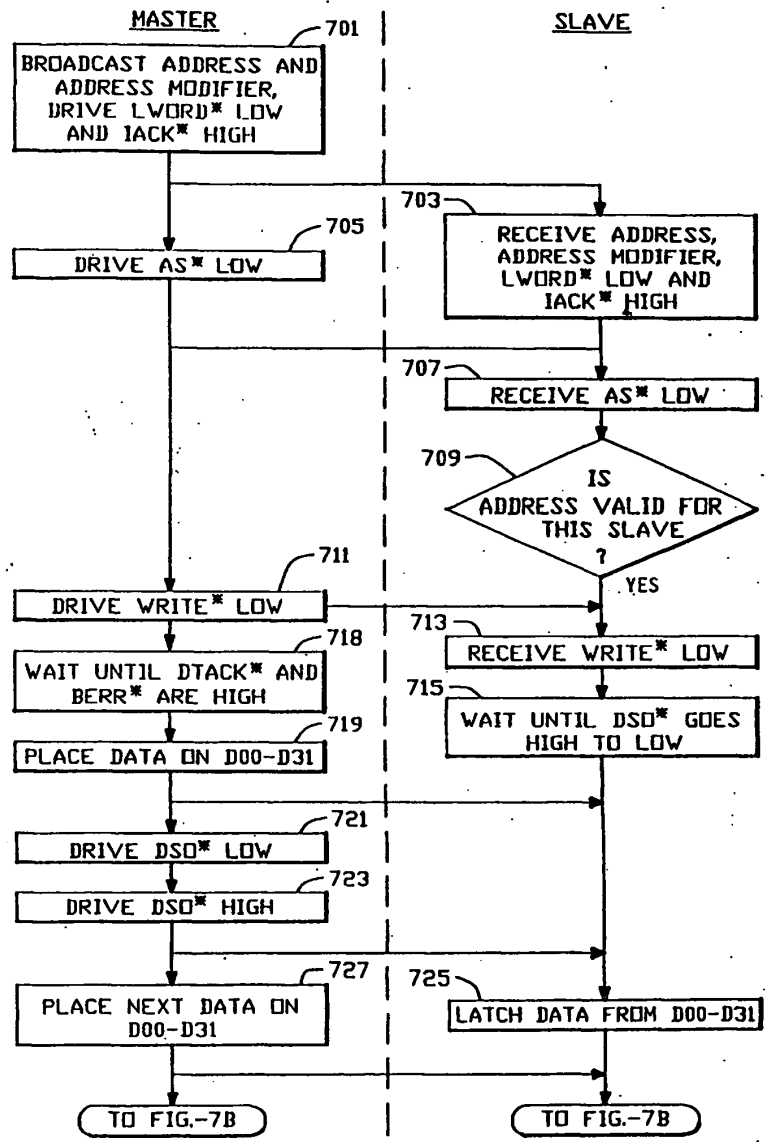


FIG.-7A

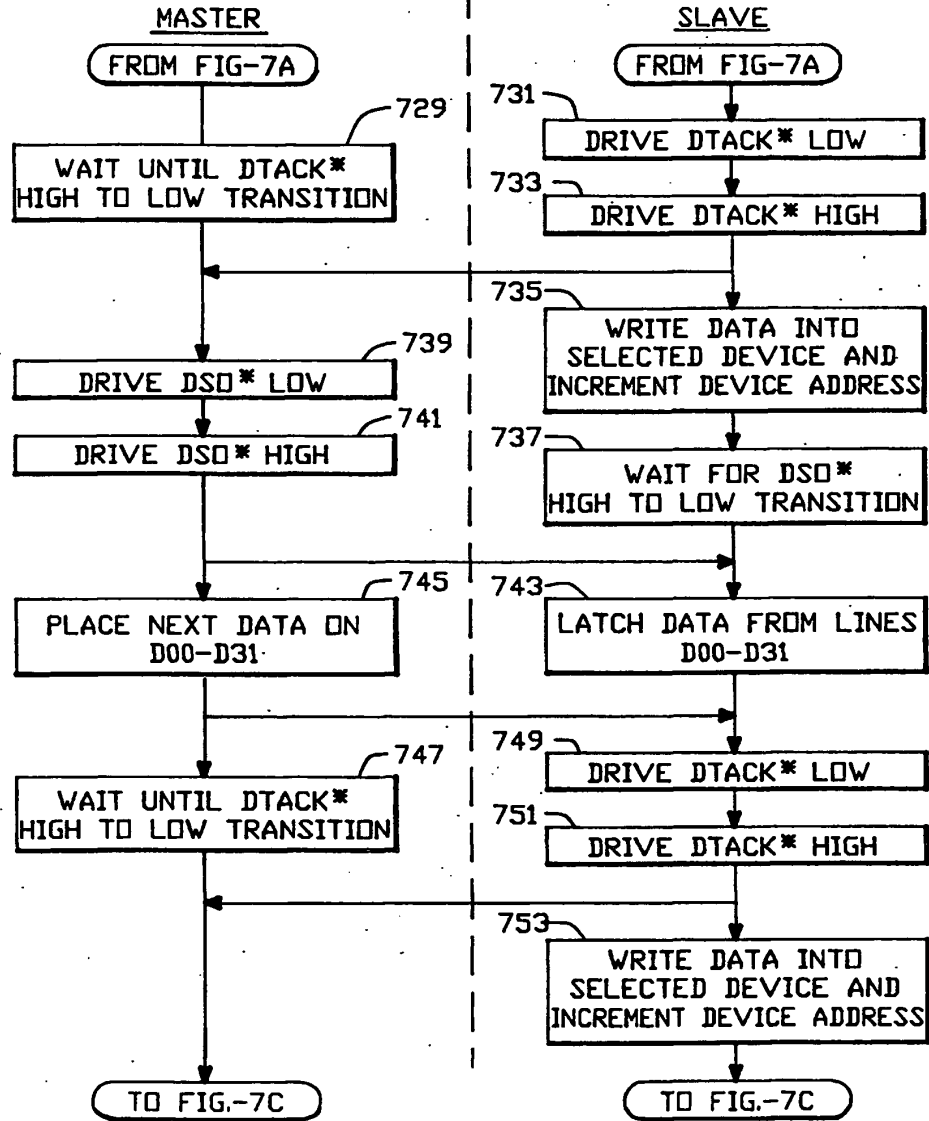


FIG.-7B



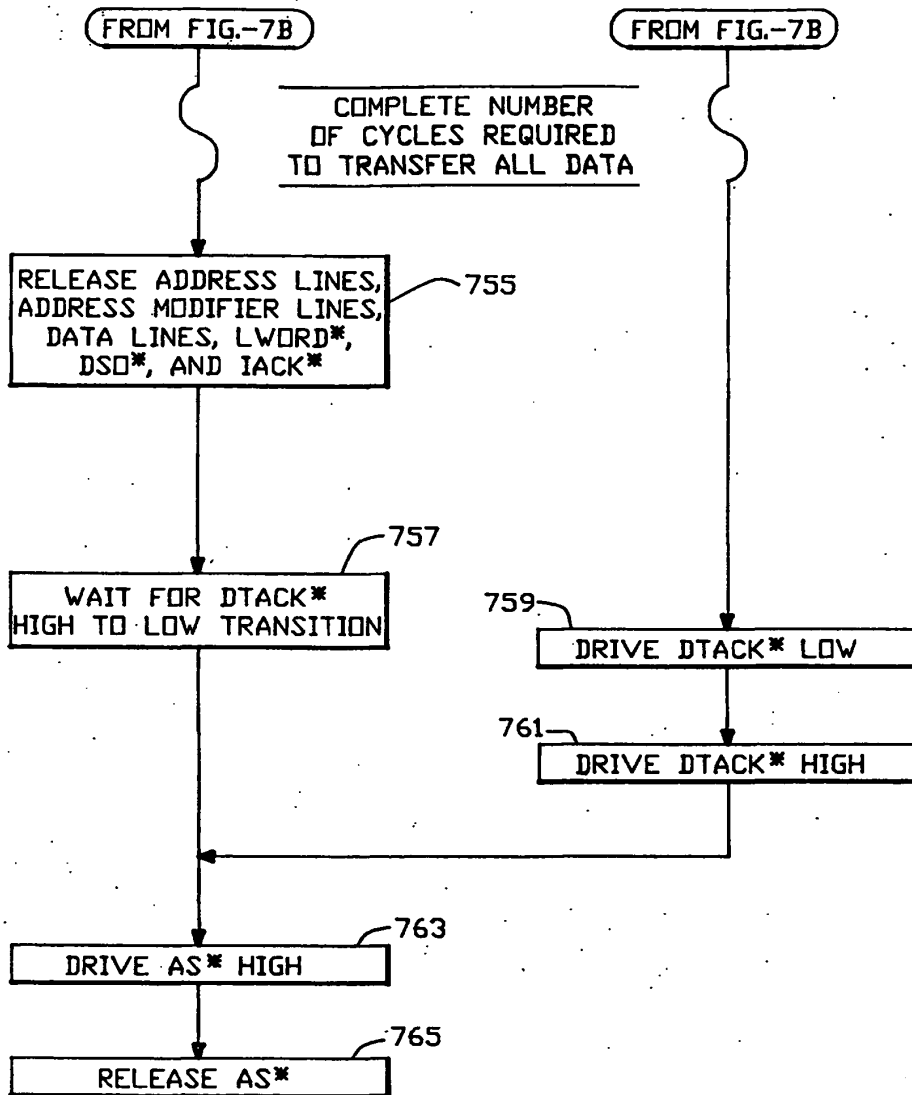


FIG.-7C

12

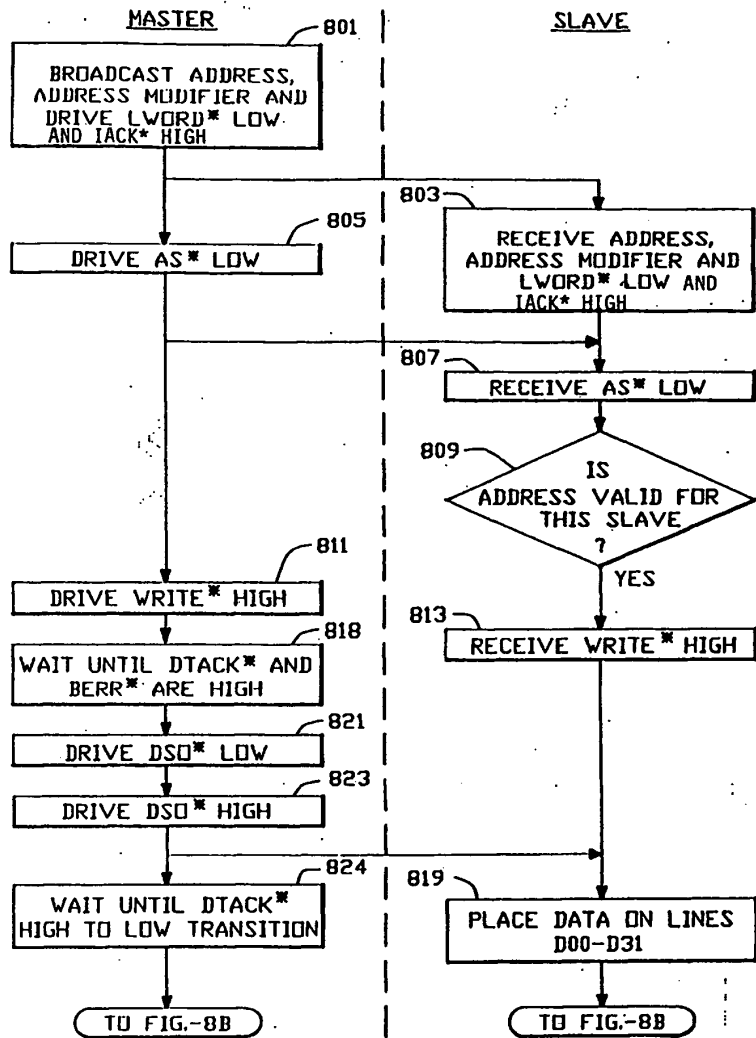


FIG.-8A

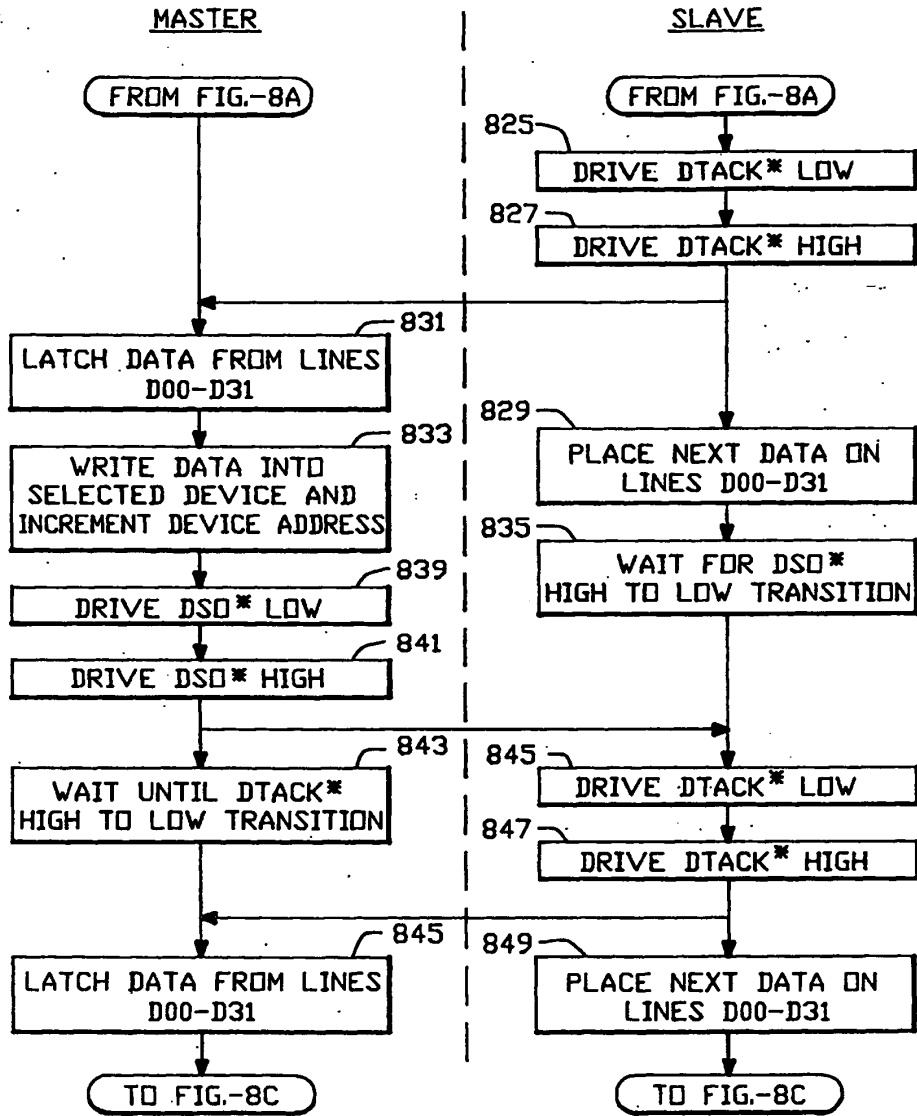


FIG.-8B

2

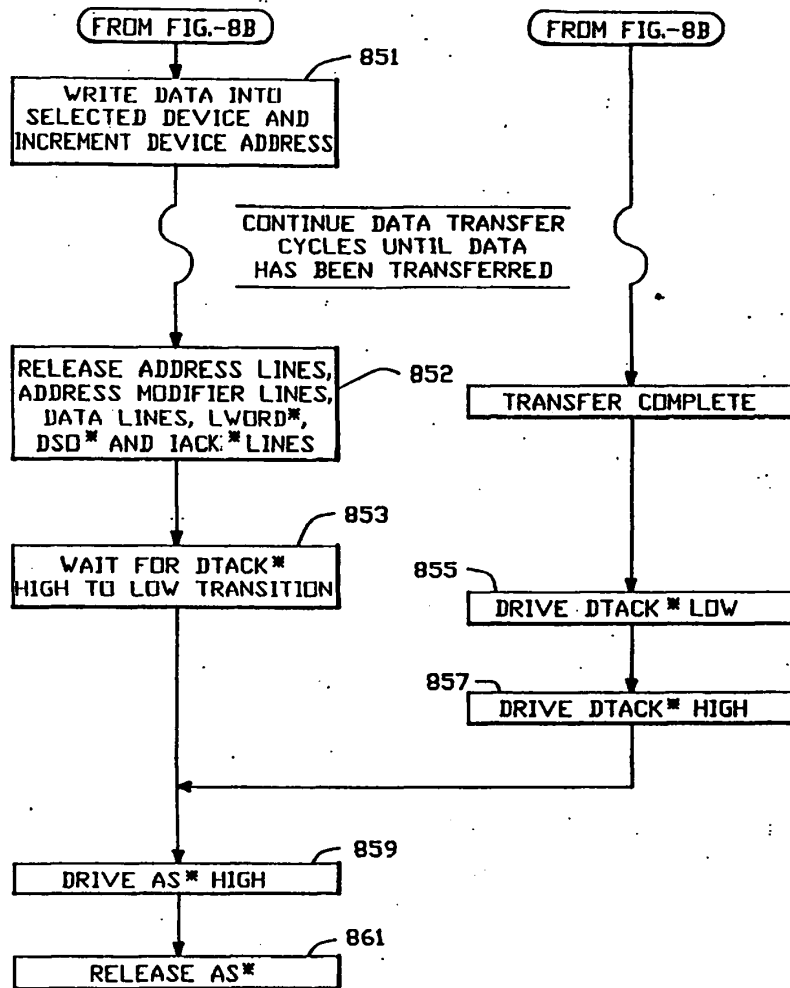


FIG.-8C



אויכטקטורה מקבילה למערכת שירות תיקים

PARALLEL I/O NETWORK FILE SERVER ARCHITECTURE

AUSPEX SYSTEMS, INC.  
C: 17980

PARALLEL I/O NETWORK FILE SERVER ARCHITECTURE

INVENTORS:

EDWARD JOHN ROW, LAURENCE B. BOUCHER,  
WILLIAM M. PITTS, STEPHEN E. BLIGHTMAN

5

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application is related to the following U.S. Patent Applications, all filed concurrently herewith:

10           1. MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE, invented by David Hitz, Allan Schwartz, James Lau and Guy Harris;

              2. ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA  
15           TRANSFER, invented by Daryl Starr; and

              3. BUS LOCKING FIFO MULTI-PROCESSOR COMMUNICATIONS SYSTEM UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER invented by Daryl D. Starr, William Pitts and Stephen Blightman.

20           The above applications are all assigned to the assignee of the present invention and are all expressly incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

25           The invention relates to computer data networks, and more particularly, to network file server architectures for computer networks.

Attorney Docket No.: AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Description of the Related Art

Over the past ten years, remarkable increases in hardware price/performance ratios have caused a startling shift in both technical and office computing environments. Distributed workstation-server networks are displacing the once pervasive dumb terminal attached to mainframe or minicomputer. To date, however, network I/O limitations have constrained the potential performance available to workstation users. This situation has developed in part because dramatic jumps in microprocessor performance have exceeded increases in network I/O performance.

In a computer network, individual user workstations are referred to as clients, and shared resources for filing, printing, data storage and wide-area communications are referred to as servers. Clients and servers are all considered nodes of a network. Client nodes use standard communications protocols to exchange service requests and responses with server nodes.

Present-day network clients and servers usually run the DOS, MacIntosh OS, OS/2, or Unix operating systems. Local networks are usually Ethernet or Token Ring at the high end, Arcnet in the midrange, or LocalTalk or StarLAN at the low end. The client-server

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7



communication protocols are fairly strictly dictated by the operating system environment -- usually one of several proprietary schemes for PCs (NetWare, 3Plus, Vines, LANManager, LANServer); AppleTalk for  
5 MacIntoshes; and TCP/IP with NFS or RFS for Unix. These protocols are all well-known in the industry.

Unix client nodes typically feature a 16- or 32-bit microprocessor with 1-8 MB of primary memory, a 640 x 1024 pixel display, and a built-in network  
10 interface. A 40-100 MB local disk is often optional. Low-end examples are 80286-based PCs or 68000-based MacIntosh I's; mid-range machines include 80386 PCs, MacIntosh II's, and 680X0-based Unix workstations; high-end machines include RISC-based DEC, HP, and Sun  
15 Unix workstations. Servers are typically nothing more than repackaged client nodes, configured in 19-inch racks rather than desk sideboxes. The extra space of a 19-inch rack is used for additional backplane slots, disk or tape drives, and power supplies.

20 Driven by RISC and CISC microprocessor developments, client workstation performance has increased by more than a factor of ten in the last few years. Concurrently, these extremely fast clients have also gained an appetite for data that remote  
25 servers are unable to satisfy. Because the I/O shortfall is most dramatic in the Unix environment, the

description of the preferred embodiment of the present invention will focus on Unix file servers. The architectural principles that solve the Unix server I/O problem, however, extend easily to server performance bottlenecks in other operating system environments as well. Similarly, the description of the preferred embodiment will focus on Ethernet implementations, though the principles extend easily to other types of networks.

10 In most Unix environments, clients and servers exchange file data using the Network File System ("NFS"), a standard promulgated by Sun Microsystems and now widely adopted by the Unix community. NFS is defined in a document entitled, "NFS: Network File System Protocol Specification," Request For Comments (RFC) 1094, by Sun Microsystems, Inc. (March 1989). This document is incorporated herein by reference in its entirety.

While simple and reliable, NFS is not optimal. Clients using NFS place considerable demands upon both networks and NFS servers supplying clients with NFS data. This demand is particularly acute for so-called diskless clients that have no local disks and therefore depend on a file server for application binaries and virtual memory paging as well as data. For these Unix client-server configurations, the ten-

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

to-one increase in client power has not been matched by a ten-to-one increase in Ethernet capacity, in disk speed, or server disk-to-network I/O throughput.

The result is that the number of diskless clients that a single modern high-end server can adequately support has dropped to between 5-10, depending on client power and application workload. For clients containing small local disks for applications and paging, referred to as dataless clients, the client-to-server ratio is about twice this, or between 10-20.

Such low client/server ratios cause piecewise network configurations in which each local Ethernet contains isolated traffic for its own 5-10 (diskless) clients and dedicated server. For overall connectivity, these local networks are usually joined together with an Ethernet backbone or, in the future, with an FDDI backbone. These backbones are typically connected to the local networks either by IP routers or MAC-level bridges, coupling the local networks together directly, or by a second server functioning as a network interface, coupling servers for all the local networks together.

In addition to performance considerations, the low client-to-server ratio creates computing problems in several additional ways:

1. Sharing. Development groups of more than 5-10 people cannot share the same server, and thus cannot easily share files without file replication and manual, multi-server updates. Bridges or routers are a partial solution but inflict a performance penalty due to more network hops.

2. Administration. System administrators must maintain many limited-capacity servers rather than a few more substantial servers. This burden includes network administration, hardware maintenance, and user account administration.

3. File System Backup. System administrators or operators must conduct multiple file system backups, which can be onerously time consuming tasks. It is also expensive to duplicate backup peripherals on each server (or every few servers if slower network backup is used).

4. Price Per Seat. With only 5-10 clients per server, the cost of the server must be shared by only a small number of users. The real cost of an entry-level Unix workstation is therefore significantly greater, often as much as 140% greater, than the cost of the workstation alone.

The widening I/O gap, as well as administrative and economic considerations, demonstrates a need for higher-performance, larger-capacity Unix file servers.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Conversion of a display-less workstation into a server may address disk capacity issues, but does nothing to address fundamental I/O limitations. As an NFS server, the one-time workstation must sustain 5-10 or more times the network, disk, backplane, and file system throughput than it was designed to support as a client. Adding larger disks, more network adaptors, extra primary memory, or even a faster processor do not resolve basic architectural I/O constraints; I/O throughput does not increase sufficiently.

Other prior art computer architectures, while not specifically designed as file servers, may potentially be used as such. In one such well-known architecture, a CPU, a memory unit, and two I/O processors are connected to a single bus. One of the I/O processors operates a set of disk drives, and if the architecture is to be used as a server, the other I/O processor would be connected to a network. This architecture is not optimal as a file server, however, at least because the two I/O processors cannot handle network file requests without involving the CPU. All network file requests that are received by the network I/O processor are first transmitted to the CPU, which makes appropriate requests to the disk-I/O processor for satisfaction of the network request.

In another such computer architecture, a disk controller CPU manages access to disk drives, and several other CPUs, three for example, may be clustered around the disk controller CPU. Each of the other CPUs can be connected to its own network. The network CPUs are each connected to the disk controller CPU as well as to each other for interprocessor communication. One of the disadvantages of this computer architecture is that each CPU in the system runs its own complete operating system. Thus, network file server requests must be handled by an operating system which is also heavily loaded with facilities and processes for performing a large number of other, non-file-server tasks. Additionally, the interprocessor communication is not optimized for file server type requests.

In yet another computer architecture, a plurality of CPUs, each having its own cache memory for data and instruction storage, are connected to a common bus with a system memory and a disk controller. The disk controller and each of the CPUs have direct memory access to the system memory, and one or more of the CPUs can be connected to a network. This architecture is disadvantageous as a file server because, among other things, both file data and the instructions for the CPUs reside in the same system memory. There will

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

be instances, therefore, in which the CPUs must stop running while they wait for large blocks of file data to be transferred between system memory and the network CPU. Additionally, as with both of the previously described computer architectures, the entire operating system runs on each of the CPUs, including the network CPU.

In yet another type of computer architecture, a large number of CPUs are connected together in a hypercube topology. One of more of these CPUs can be connected to networks, while another can be connected to disk drives. This architecture is also disadvantageous as a file server because, among other things, each processor runs the entire operating system. Interprocessor communication is also not optimal for file server applications.

#### SUMMARY OF THE INVENTION

The present invention involves a new, server-specific I/O architecture that is optimized for a Unix file server's most common actions -- file operations. Roughly stated, the invention involves a file server architecture comprising one or more network controllers, one or more file controllers, one or more storage processors, and a system or buffer memory, all connected over a message passing bus and operating in

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSE/7209.001

8/24/89-7

parallel with the Unix host processor. The network controllers each connect to one or more network, and provide all protocol processing between the network layer data format and an internal file server format  
5 for communicating client requests to other processors in the server. Only those data packets which cannot be interpreted by the network controllers, for example client requests to run a client-defined program on the server, are transmitted to the Unix host for  
10 processing. Thus the network controllers, file controllers and storage processors contain only small parts of an overall operating system, and each is optimized for the particular type of work to which it is dedicated.

15 Client requests for file operations are transmitted to one of the file controllers which, independently of the Unix host, manages the virtual file system of a mass storage device which is coupled to the storage processors. The file controllers may  
20 also control data buffering between the storage processors and the network controllers, through the system memory. The file controllers preferably each include a local buffer memory for caching file control information, separate from the system memory for  
25 caching file data. Additionally, the network controllers, file processors and storage processors are



all designed to avoid any instruction fetches from the system memory, instead keeping all instruction memory separate and local. This arrangement eliminates contention on the backplane between microprocessor instruction fetches and transmissions of message and file data.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be described with respect to particular embodiments thereof, and reference will be made to the drawings, in which:

Fig. 1. is a block diagram of a prior art file server architecture;

Fig. 2 is a block diagram of a file server architecture according to the invention;

Fig. 3 is a block diagram of one of the network controllers shown in Fig. 2;

Fig. 4 is a block diagram of one of the file controllers shown in Fig. 2;

Fig. 5 is a block diagram of one of the storage processors shown in Fig. 2;

Fig. 6 is a block diagram of one of the system memory cards shown in Fig. 2;

Figs. 7A-C are a flowchart illustrating the operation of a fast transfer protocol BLOCK WRITE cycle; and

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Figs. 8A-C are a flowchart illustrating the operation of a fast transfer protocol BLOCK READ cycle.

DETAILED DESCRIPTION

5 For comparison purposes and background, an illustrative prior-art file server architecture will first be described with respect to Fig. 1. Fig. 1 is an overall block diagram of a conventional prior-art Unix-based file server for Ethernet networks. It  
10 consists of a host CPU card 10 with a single microprocessor on board. The host CPU card 10 connects to an Ethernet #1 12, and it connects via a memory management unit (MMU) 11 to a large memory array 16. The host CPU card 10 also drives a keyboard, a video  
15 display, and two RS232 ports (not shown). It also connects via the MMU 11 and a standard 32-bit VME bus 20 to various peripheral devices, including an SMD disk controller 22 controlling one or two disk drives 24, a SCSI host adaptor 26 connected to a SCSI bus 28, a  
20 tape controller 30 connected to a quarter-inch tape drive 32, and possibly a network #2 controller 34 connected to a second Ethernet 36. The SMD disk controller 22 can communicate with memory array 16 by direct memory access via bus 20 and MMU 11, with either  
25 the disk controller or the MMU acting as a bus master.

This configuration is illustrative; many variations are available.

The system communicates over the Ethernets using industry standard TCP/IP and NFS protocol stacks. A description of protocol stacks in general can be found in Tanenbaum, "Computer Networks" (Second Edition, Prentice Hall: 1988). File server protocol stacks are described at pages 535-546. The Tanenbaum reference is incorporated herein by reference.

Basically, the following protocol layers are implemented in the apparatus of Fig. 1:

Network Layer. The network layer converts data packets between a format specific to Ethernets and a format which is independent of the particular type of network used. the Ethernet-specific format which is used in the apparatus of Fig. 1 is described in Hornig, "A Standard For The Transmission of IP Datagrams Over Ethernet Networks," RFC 894 (April 1984), which is incorporated herein by reference.

The Internet Protocol (IP) Layer. This layer provides the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. For messages to be sent from the file server to a client, a higher level in the server calls the IP module, providing the internet address of the

destination client and the message to transmit. The IP module performs any required fragmentation of the message to accommodate packet size limitations of any intervening gateway, adds internet headers to each  
5 fragment, and calls on the network layer to transmit the resulting internet datagrams. The internet header includes a local network destination address (translated from the internet address) as well as other parameters.

10 For messages received by the IP layer from the network layer, the IP module determines from the internet address whether the datagram is to be forwarded to another host on another network, for example on a second Ethernet such as 36 in Fig. 1, or  
15 whether it is intended for the server itself. If it is intended for another host on the second network, the IP module determines a local net address for the destination and calls on the local network layer for that network to send the datagram. If the datagram is  
20 intended for an application program within the server, the IP layer strips off the header and passes the remaining portion of the message to the appropriate next higher layer. The internet protocol standard used  
25 in the illustrative apparatus of Fig. 1 is specified in Information Sciences Institute, "Internet Protocol, DARPA Internet Program Protocol Specification," RFC 791

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

(September 1981), which is incorporated herein by reference.

TCP/UDP Layer. This layer is a datagram service with more elaborate packaging and addressing options than the IP layer. For example, whereas an IP datagram can hold about 1,500 bytes and be addressed to hosts, UDP datagrams can hold about 64KB and be addressed to a particular port within a host. TCP and UDP are alternative protocols at this layer; applications requiring ordered reliable delivery of streams of data may use TCP, whereas applications (such as NFS) which do not require ordered and reliable delivery may use UDP.

The prior art file server of Fig. 1 uses both TCP and UDP. It uses UDP for file server-related services, and uses TCP for certain other services which the server provides to network clients. The UDP is specified in Postel, "User Datagram Protocol," RFC 768 (August 28, 1980), which is incorporated herein by reference. TCP is specified in Postel, "Transmission Control Protocol," RFC 761 (January 1980) and RFC 793 (September 1981), which is also incorporated herein by reference.

XDR/RPC Layer. This layer provides functions callable from higher level programs to run a designated procedure on a remote machine. It also provides the

5 decoding necessary to permit a client machine to  
execute a procedure on the server. For example, a  
caller process in a client node may send a call message  
to the server of Fig. 1. The call message includes a  
specification of the desired procedure, and its  
parameters. The message is passed up the stack to the  
RPC layer, which calls the appropriate procedure within  
the server. When the procedure is complete, a reply  
message is generated and RPC passes it back down the  
stack and over the network to the caller client. RPC  
10 is described in Sun Microsystems, Inc., "RPC: Remote  
Procedure Call Protocol Specification, Version 2," RFC  
1057 (June 1988), which is incorporated herein by  
reference.

15 RPC uses the XDR external data representation  
standard to represent information passed to and from  
the underlying UDP layer. XDR is merely a data  
encoding standard, useful for transferring data between  
different computer architectures. Thus, on the network  
20 side of the XDR/RPC layer, information is machine-  
independent; on the host application side, it may not  
be. XDR is described in Sun Microsystems, Inc., "XDR:  
External Data Representation Standard," RFC 1014 (June  
1987), which is incorporated herein by reference.

25 NFS Layer. The NFS ("network file system")  
layer is one of the programs available on the server

which an RPC request can call. The combination of host address, program number, and procedure number in an RPC request can specify one remote NFS procedure to be called.

5 Remote procedure calls to NFS on the file server of Fig. 1 provide transparent, stateless, remote access to shared files on the disks 24. NFS assumes a file system that is hierarchical, with directories as all but the bottom level of files. Client hosts can call  
10 any of about 20 NFS procedures including such procedures as reading a specified number of bytes from a specified file; writing a specified number of bytes to a specified file; creating, renaming and removing specified files; parsing directory trees; creating and  
15 removing directories; and reading and setting file attributes. The location on disk to which and from which data is stored and retrieved is always specified in logical terms, such as by a file handle or Inode designation and a byte offset. The details of the  
20 actual data storage are hidden from the client. The NFS procedures, together with possible higher level modules such as Unix VFS and UFS, perform all conversion of logical data addresses to physical data addresses such as drive, head, track and sector  
25 identification. NFS is specified in Sun Microsystems, Inc., "NFS: Network File System Protocol

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Specification," RFC 1094 (March 1989), incorporated herein by reference.

5 With the possible exception of the network layer, all the protocol processing described above is done in software, by a single processor in the host CPU card 10. That is, when an Ethernet packet arrives on Ethernet 12, the host CPU 10 performs all the protocol processing in the NFS stack, as well as the protocol processing for any other application which may be 10 running on the host 10. NFS procedures are run on the host CPU 10, with access to memory 16 for both data and program code being provided via MMU 11. Logically specified data addresses are converted to a much more physically specified form and communicated to the SMD 15 disk controller 22 or the SCSI bus 28, via the VME bus 20, and all disk caching is done by the host CPU 10 through the memory 16. The host CPU card 10 also runs procedures for performing various other functions of the file server, communicating with tape controller 30 20 via the VME bus 20. Among these are client-defined remote procedures requested by client workstations.

If the server serves a second Ethernet 36, packets from that Ethernet are transmitted to the host CPU 10 over the same VME bus 20 in the form of IP datagrams. 25 Again, all protocol processing except for the network layer is performed by software processes running on the



host CPU 10. In addition, the protocol processing for any message that is to be sent from the server out on either of the Ethernets 12 or 36 is also done by processes running on the host CPU 10.

5           It can be seen that the host CPU 10 performs an enormous amount of processing of data, especially if 5-10 clients on each of the two Ethernets are making file server requests and need to be sent responses on a frequent basis. The host CPU 10 runs a multitasking  
10 Unix operating system, so each incoming request need not wait for the previous request to be completely processed and returned before being processed. Multiple processes are activated on the host CPU 10 for performing different stages of the processing of  
15 different requests, so many requests may be in process at the same time. But there is only one CPU on the card 10, so the processing of these requests is not accomplished in a truly parallel manner. The processes are instead merely time sliced. The CPU 10 therefore  
20 represents a major bottleneck in the processing of file server requests.

          Another bottleneck occurs in MMU 11, which must transmit both instructions and data between the CPU card 10 and the memory 16. All data flowing between  
25 the disk drives and the network passes through this interface at least twice.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Yet another bottleneck can occur on the VME bus 20, which must transmit data among the SMD disk controller 22, the SCSI host adaptor 26, the host CPU card 10, and possibly the network #2 controller 34.

5 PREFERRED EMBODIMENT-OVERALL HARDWARE ARCHITECTURE

In Fig. 2 there is shown a block diagram of a network file server 100 according to the invention. It can include multiple network controller (NC) boards, one or more file controller (FC) boards, one or more  
10 storage processor (SP) boards, multiple system memory boards, and one or more host processors. The particular embodiment shown in Fig. 2 includes four network controller boards 110a-110d, two file controller boards 112a-112b, two storage processors  
15 114a-114b, four system memory cards 116a-116d for a total of 192MB of memory, and one local host processor 118. The boards 110, 112, 114, 116 and 118 are connected together over a VME bus 120 on which an enhanced block transfer mode as described in the  
20 ENHANCED VMEBUS PROTOCOL application identified above may be used. Each of the four network controllers 110 shown in Fig. 2 can be connected to up to two Ethernets 122, for a total capacity of 8 Ethernets 122a-122h. Each of the storage processors 114 operates ten  
25 parallel SCSI busses, nine of which can each support up

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

to three SCSI disk drives each. The tenth SCSI channel on each of the storage processors 114 is used for tape drives and other SCSI peripherals.

The host 118 is essentially a standard SunOs Unix processor, providing all the standard Sun Open Network Computing (ONC) services except NFS and IP routing. Importantly, all network requests to run a user-defined procedure are passed to the host for execution. Each of the NC boards 110, the FC boards 112 and the SP boards 114 includes its own independent 32-bit microprocessor. These boards essentially off-load from the host processor 118 virtually all of the NFS and disk processing. Since the vast majority of messages to and from clients over the Ethernets 122 involve NFS requests and responses, the processing of these requests in parallel by the NC, FC and SP processors, with minimal involvement by the local host 118, vastly improves file server performance. Unix is explicitly eliminated from virtually all network, file, and storage processing.

OVERALL SOFTWARE ORGANIZATION AND DATA FLOW

Prior to a detailed discussion of the hardware subsystems shown in Fig. 2, an overview of the software structure will now be undertaken. The software organization is described in more detail in the above-

identified application entitled MULTIPLE FACILITY  
OPERATING SYSTEM ARCHITECTURE.

Most of the elements of the software are well  
known in the field and are found in most networked Unix  
5 systems, but there are two components which are not:  
Local NFS ("LNFS") and the messaging kernel ("MK")  
operating system kernel. These two components will be  
explained first.

The Messaging Kernel. The various processors in  
10 file server 100 communicate with each other through the  
use of a messaging kernel running on each of the  
processors 110, 112, 114 and 118. These processors do  
not share any instruction memory, so task-level  
communication cannot occur via straightforward  
15 procedure calls as it does in conventional Unix.  
Instead, the messaging kernel passes messages over VME  
bus 120 to accomplish all necessary inter-processor  
communication. Message passing is preferred over  
remote procedure calls for reasons of simplicity and  
20 speed.

Messages passed by the messaging kernel have a  
fixed 128-byte length. Within a single processor,  
messages are sent by reference; between processors,  
they are copied by the messaging kernel and then  
25 delivered to the destination process by reference. The  
processors of Fig. 2 have special hardware, discussed

below, that can expediently exchange and buffer inter-processor messaging kernel messages.

The LNFS Local NFS interface. The 22-function NFS standard was specifically designed for stateless operation using unreliable communication. This means that neither clients nor server can be sure if they hear each other when they talk (unreliability). In practice, in an Ethernet environment, this works well.

5  
10        Within the server 100, however, NFS level datagrams are also used for communication between processors, in particular between the network controllers 110 and the file controller 112, and between the host processor 118 and the file controller  
15 112. For this internal communication to be both efficient and convenient, it is undesirable and impractical to have complete statelessness or unreliable communications. Consequently, a modified form of NFS, namely LNFS, is used for internal  
20 communication of NFS requests and responses. LNFS is used only within the file server 100; the external network protocol supported by the server is precisely standard, licensed NFS. LNFS is described in more detail below.

25        The Network Controllers 110 each run an NFS server which, after all protocol processing is done up to the

NFS layer, converts between external NFS requests and responses and internal LNFS requests and responses. For example, NFS requests arrive as RPC requests with XDR and enclosed in a UDP datagram. After protocol  
5 processing, the NFS server translates the NFS request into LNFS form and uses the messaging kernel to send the request to the file controller 112.

The file controller runs an LNFS server which handles LNFS requests both from network controllers and  
10 from the host 118. The LNFS server translates LNFS requests to a form appropriate for a file system server, also running on the file controller, which manages the system memory file data cache through a block I/O layer.

15 An overview of the software in each of the processors will now be set forth.

#### Network Controller 110

The optimized dataflow of the server 100 begins with the intelligent network controller 110. This  
20 processor receives Ethernet packets from client workstations. It quickly identifies NFS-destined packets and then performs full protocol processing on them to the NFS level, passing the resulting LNFS requests directly to the file controller 112. This  
25 protocol processing includes IP routing and reassembly,

UDP demultiplexing, XDR decoding, and NFS request dispatching. The reverse steps are used to send an NFS reply back to a client. Importantly, these time-consuming activities are performed directly in the Network Controller 110, not in the host 118.

The server 100 uses conventional NFS ported from Sun Microsystems, Inc., Mountain View, CA, and is NFS protocol compatible.

Non-NFS network traffic is passed directly to its destination host processor 118.

The NCs 110 also perform their own IP routing. Each network controller 110 supports two fully parallel Ethernets. There are four network controllers in the embodiment of the server 100 shown in Fig. 2, so that server can support up to eight Ethernets. For the two Ethernets on the same network controller 110, IP routing occurs completely within the network controller and generates no backplane traffic. Thus attaching two mutually active Ethernets to the same controller not only minimizes their inter-net transit time, but also significantly reduces backplane contention on the VME bus 120. Routing table updates are distributed to the network controllers from the host processor 118, which runs either the gated or routed Unix demon.

While the network controller described here is designed for Ethernet LANs, it will be understood that the invention can be used just as readily with other network types, including FDDI.

5           File Controller 112

In addition to dedicating a separate processor for NFS protocol processing and IP routing, the server 100 also dedicates a separate processor, the intelligent file controller 112, to be responsible for all file system processing. It uses conventional Berkeley Unix 10 4.3 file system code and uses a binary-compatible data representation on disk. These two choices allow all standard file system utilities (particularly block-level tools) to run unchanged.

15           The file controller 112 runs the shared file system used by all NCs 110 and the host processor 118. Both the NCs and the host processor communicate with the file controller 112 using the LNFS interface. The NCs 110 use LNFS as described above, while the host 20 processor 118 uses LNFS as a plug-in module to SunOs's standard Virtual File System ("VFS") interface.

When an NC receives an NFS read request from a client workstation, the resulting LNFS request passes to the FC 112. The FC 112 first searches the system 25 memory 116 buffer cache for the requested data. If



found, a reference to the buffer is returned to the NC  
110. If not found, the LRU (least recently used) cache  
buffer in system memory 116 is freed and reassigned for  
the requested block. The FC then directs the SP 114 to  
5 read the block into the cache buffer from a disk drive  
array. When complete, the SP so notifies the FC, which  
in turn notifies the NC 110. The NC 110 then sends an  
NFS reply, with the data from the buffer, back to the  
NFS client workstation out on the network. Note that  
10 the SP 114 transfers the data into system memory 116,  
if necessary, and the NC 110 transfers the data from  
system memory 116 to the networks. The process takes  
place without any involvement of the host 118.

Storage Processor

15 The intelligent storage processor 114 manages all  
disk and tape storage operations. While autonomous,  
storage processors are primarily directed by the file  
controller 112 to move file data between system memory  
116 and the disk subsystem. The exclusion of both the  
20 host 118 and the FC 112 from the actual data path helps  
to supply the performance needed to service many  
remote clients.

Additionally, coordinated by a Server Manager in  
the host 118, storage processor 114 can execute server  
25 backup by moving data between the disk subsystem and

Attorney Docket No.:AUSP7209  
WPI/WSW/AUSP/7209.001

8/24/89-7

tape or other archival peripherals on the SCSI channels. Further, if directly accessed by host processor 118, SP 114 can provide a much higher performance conventional disk interface for Unix, virtual memory, and databases. In Unix nomenclature, the host processor 118 can mount boot, storage swap, and raw partitions via the storage processors 114.

Each storage processor 114 operates ten parallel, fully synchronous SCSI channels (busses) simultaneously. Nine of these channels support three arrays of nine SCSI disk drives each, each drive in an array being assigned to a different SCSI channel. The tenth SCSI channel hosts up to seven tape and other SCSI peripherals. In addition to performing reads and writes, SP 114 performs device-level optimizations such as disk seek queue sorting, directs device error recovery, and controls DMA transfers between the devices and system memory 116.

#### Host Processor 118

The local host 118 has three main purposes: to run Unix, to provide standard ONC network services for clients, and to run a Server Manager. Since Unix and ONC are ported from the standard SunOs Release 4 and ONC Services Release 2, the server 100 can provide identically compatible high-level ONC services such as

the Yellow Pages, Lock Manager, DES Key Authenticator, Auto Mounter, and Port Mapper. Sun/2 Network disk booting and more general IP internet services such as Telnet, FTP, SMTP, SNMP, and reverse ARP are also supported. Finally, print spoolers and similar Unix demons operate transparently.

The host processor 118 runs the following software modules:

TCP and socket layers. The Transport Control Protocol ("TCP"), which is used for certain server functions other than NFS, provides reliable bytestream communication between two processors. Sockets are used to establish TCP connections.

VFS interface. The Virtual File System ("VFS") interface is a standard SunOs file system interface. It paints a uniform file-system picture for both users and the non-file parts of the Unix operating system, hiding the details of the specific file system. Thus standard NFS, LNFS, and any local Unix file system can coexist harmoniously.

UFS interface. The Unix File System ("UFS") interface is the traditional and well-known Unix interface for communication with local-to-the-processor disk drives. In the server 100, it is used to occasionally mount storage processor volumes directly, without going through the file controller 112.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Normally, the host 118 uses LNFS and goes through the file controller.

5        Device layer. The device layer is a standard software interface between the Unix device model and different physical device implementations. In the server 100, disk devices are not attached to host processors directly, so the disk driver in the host's device layer uses the messaging kernel to communicate with the storage processor 114.

10       Route and Port Mapper Demons. The Route and Port Mapper demons are Unix user-level background processes that maintain the Route and Port databases for packet routing. They are mostly inactive and not in any performance path.

15       Yellow Pages and Authentication Demon. The Yellow Pages and Authentication services are Sun-ONC standard network services. Yellow Pages is a widely used multipurpose name-to-name directory lookup service. The Authentication service uses cryptographic keys to  
20       authenticate, or validate, requests to insure that requestors have the proper privileges for any actions or data they desire.

25       Server Manager. The Server Manager is an administrative application suite that controls configuration, logs error and performance reports, and provides a monitoring and tuning interface for the

system administrator. These functions can be exercised from either system console connected to the host 118, or from a system administrator's workstation.

The host processor 118 is a conventional OEM Sun central processor card, Model 3E/120. It incorporates a Motorola 68020 microprocessor and 4MB of on-board memory. Other processors, such as a SPARC-based processor, are also possible.

The structure and operation of each of the hardware components of server 100 will now be described in detail.

#### NETWORK CONTROLLER HARDWARE ARCHITECTURE

Fig. 3 is a block diagram showing the data path and some control paths for an illustrative one of the network controllers 110a. It comprises a 20 MHz 68020 microprocessor 210 connected to a 32-bit microprocessor data bus 212. Also connected to the microprocessor data bus 212 is a 256K byte CPU memory 214. The low order 8 bits of the microprocessor data bus 212 are connected through a bidirectional buffer 216 to an 8-bit slow-speed data bus 218. On the slow-speed data bus 218 is a 128K byte EPROM 220, a 32 byte PROM 222, and a multi-function peripheral (MFP) 224. The EPROM 220 contains boot code for the network controller 110a, while the PROM 222 stores various operating parameters

such as the Ethernet addresses assigned to each of the two Ethernet interfaces on the board. Ethernet address information is read into the corresponding interface control block in the CPU memory 214 during initialization. The MFP 224 is a Motorola 68901, and performs various local functions such as timing, interrupts, and general purpose I/O. The MFP 224 also includes a UART for interfacing to an RS232 port 226. These functions are not critical to the invention and will not be further described herein.

The low order 16 bits of the microprocessor data bus 212 are also coupled through a bidirectional buffer 230 to a 16-bit LAN data bus 232. A LAN controller chip 234, such as the Am7990 LANCE Ethernet controller manufactured by Advanced Micro Devices, Inc. Sunnyvale, CA., interfaces the LAN data bus 232 with the first Ethernet 122a shown in Fig. 2. Control and data for the LAN controller 234 are stored in a 512K byte LAN memory 236, which is also connected to the LAN data bus 232. A specialized 16 to 32 bit FIFO chip 240, referred to herein as a parity FIFO chip and described below, is also connected to the LAN data bus 232. Also connected to the LAN data bus 232 is a LAN DMA controller 242, which controls movements of packets of data between the LAN memory 236 and the FIFO chip 240.

The LAN DMA controller 242 may be a Motorola M68440 DMA controller using channel zero only.

5 The second Ethernet 122b shown in Fig. 2 connects to a second LAN data bus 252 on the network controller card 110a shown in Fig. 3. The LAN data bus 252 connects to the low order 16 bits of the microprocessor data bus 212 via a bidirectional buffer 250, and has similar components to those appearing on the LAN data bus 232. In particular, a LAN controller 10 254 interfaces the LAN data bus 252 with the Ethernet 122b, using LAN memory 256 for data and control, and a LAN DMA controller 262 controls DMA transfer of data between the LAN memory 256 and the 16-bit wide data port A of the parity FIFO 260.

15 The low order 16 bits of microprocessor data bus 212 are also connected directly to another parity FIFO 270, and also to a control port of a VME/FIFO DMA controller 272. The FIFO 270 is used for passing messages between the CPU memory 214 and one of the 20 remote boards 110, 112, 114, 116 or 118 (Fig. 2) in a manner described below. The VME/FIFO DMA controller 272, which supports three round-robin non-prioritized channels for copying data, controls all data transfers between one of the remote boards and any of the FIFOs 25 240, 260 or 270, as well as between the FIFOs 240 and 260.

32-bit data bus 274, which is connected to the 32-bit port B of each of the FIFOs 240, 260 and 270, is the data bus over which these transfers take place. Data bus 274 communicates with a local 32-bit bus 276 via a bidirectional pipelining latch 278, which is also controlled by VME/FIFO DMA controller 272 which in turn communicates with the VME bus 120 via a bidirectional buffer 280.

The local data bus 276 is also connected to a set of control registers 282, which are directly addressable across the VME bus 120. The registers 282 are used mostly for system initialization and diagnostics.

The local data bus 276 is also coupled to the microprocessor data bus 212 via a bidirectional buffer 284. When the NC 110a operates in slave mode, the CPU memory 214 is directly addressable from VME bus 120. One of the remote boards can copy data directly from the CPU memory 214 via the bidirectional buffer 284. LAN memories 236 and 256 are not directly addressed over VME bus 120.

The parity FIFOs 240, 260 and 270 each consist of an ASIC, the functions and operation of which are described in the Appendix. The FIFOs 240 and 260 are configured for packet data transfer and the FIFO 270 is configured for message passing. Referring to the



Appendix C, the FIFOs 240 and 260 are programmed with the following bit settings in the Data Transfer Configuration Register:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
5	0	WD Mode	N/A
	1	Parity Chip	N/A
	2	Parity Correct Mode	N/A
	3	8/16 bits CPU & PortA interface	16 bits (1)
	4	Invert Port A address 0	no (0)
10	5	Invert Port A address 1	yes (1)
	6	Checksum Carry Wrap	yes (1)
	7	Reset	no (0)

The Data Transfer Control Register is programmed

as follows:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
15	0	Enable PortA Req/Ack	yes (1)
	1	Enable PortB Req/Ack	yes (1)
	2	Data Transfer Direction	(as desired)
	3	CPU parity enable	no (0)
20	4	PortA parity enable	no (0)
	5	PortB parity enable	no (0)
	6	Checksum Enable	yes (1)
	7	PortA Master	yes (1)

Unlike the configuration used on FIFOs 240 and 260, the microprocessor 210 is responsible for loading and unloading Port A directly. The microprocessor 210

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7.

reads an entire 32-bit word from port A with a single instruction using two port A access cycles. Port A data transfer is disabled by unsetting bits 0 (Enable PortA Req/Ack) and 7 (PortA Master) of the Data Transfer Control Register.

The remainder of the control settings in FIFO 270 are the same as those in FIFOs 240 and 260 described above.

The NC 110a also includes a command FIFO 290. The command FIFO 290 includes an input port coupled to the local data bus 276, and which is directly addressable across the VME bus 120, and includes an output port connected to the microprocessor data bus 212. As explained in more detail below, when one of the remote boards issues a command or response to the NC 110a, it does so by directly writing a 1-word (32-bit) message descriptor into NC 110a's command FIFO 290. Command FIFO 290 generates a "FIFO not empty" status to the microprocessor 210, which then reads the message descriptor off the top of FIFO 290 and processes it. If the message is a command, then it includes a VME address at which the message is located (presumably an address in a shared memory similar to 214 on one of the remote boards). The microprocessor 210 then programs the FIFO 270 and the VME/FIFO DMA controller 272 to

copy the message from the remote location into the CPU memory 214.

5 Command FIFO 290 is a conventional two-port FIFO, except that additional circuitry is included for generating a Bus Error signal on VME bus 120 if an attempt is made to write to the data input port while the FIFO is full. Command FIFO 290 has space for 256 entries.

10 A noteworthy feature of the architecture of NC 110a is that the LAN buses 232 and 252 are independent of the microprocessor data bus 212. Data packets being routed to or from an Ethernet are stored in LAN memory 236 on the LAN data bus 232 (or 256 on the LAN data bus 252), and not in the CPU memory 214. Data transfer  
15 between the LAN memories 236 and 256 and the Ethernets 122a and 122b, are controlled by LAN controllers 234 and 254, respectively, while most data transfer between LAN memory 236 or 256 and a remote port on the VME bus 120 are controlled by LAN DMA controllers 242 and 262,  
20 FIFOs 240 and 260, and VME/FIFO DMA controller 272. An exception to this rule occurs when the size of the data transfer is small, e.g., less than 64 bytes, in which case microprocessor 210 copies it directly without using DMA. The microprocessor 210 is not involved in  
25 larger transfers except in initiating them and in receiving notification when they are complete.

The CPU memory 214 contains mostly instructions for microprocessor 210, messages being transmitted to or from a remote board via FIFO 270, and various data blocks for controlling the FIFOs, the DMA controllers and the LAN controllers. The microprocessor 210 accesses the data packets in the LAN memories 236 and 256 by directly addressing them through the bidirectional buffers 230 and 250, respectively, for protocol processing. The local high-speed static RAM in CPU memory 214 can therefore provide zero wait state memory access for microprocessor 210 independent of network traffic. This is in sharp contrast to the prior art architecture shown in Fig. 1, in which all data and data packets, as well as microprocessor instructions for host CPU card 10, reside in the memory 16 and must communicate with the host CPU card 10 via the MMU 11.

While the LAN data buses 232 and 252 are shown as separate buses in Fig. 3, it will be understood that they may instead be implemented as a single combined bus.

NETWORK CONTROLLER OPERATION

In operation, when one of the LAN controllers (such as 234) receives a packet of information over its Ethernet 122a, it reads in the entire packet and stores

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

it in corresponding LAN memory 236. The LAN controller  
234 then issues an interrupt to microprocessor 210 via  
MFP 224, and the microprocessor 210 examines the status  
register on LAN controller 234. (via bidirectional  
5 buffer 230) to determine that the event causing the  
interrupt was a "receive packet completed." In order  
to avoid a potential lockout of the second Ethernet  
122b caused by the prioritized interrupt handling  
characteristic of MFP 224, the microprocessor 210 does  
10 not at this time immediately process the received  
packet; instead, such processing is scheduled for a  
polling function.

When the polling function reaches the processing  
of the received packet, control over the packet is  
15 passed to a software link level receive module. The  
link level receive module then decodes the packet  
according to either of two different frame formats:  
standard Ethernet format or SNAP (IEEE 802 LCC) format.  
An entry in the header in the packet specifies which  
20 frame format was used. The link level driver then  
determines which of three types of messages is  
contained in the received packet: (1) IP, (2) ARP  
packets which can be handled by a local ARP module, or  
(3) ARP packets and other packet types which must be  
25 forwarded to the local host 118 (Fig. 2) for  
processing. If the packet is an ARP packet which can

be handled by the NC 110a, such as a request for the address of server 100, then the microprocessor 210 assembles a response packet in LAN memory 236 and, in a conventional manner, causes LAN controller 234 to transmit that packet back over Ethernet 122a. It is noteworthy that the data manipulation for accomplishing this task is performed almost completely in LAN memory 236, directly addressed by microprocessor 210 as controlled by instructions in CPU memory 214. The function is accomplished also without generating any traffic on the VME backplane 120 at all, and without disturbing the local host 118.

If the received packet is either an ARP packet which cannot be processed completely in the NC 110a, or is another type of packet which requires delivery to the local host 118 (such as a client request for the server 100 to execute a client-defined procedure), then the microprocessor 210 programs LAN DMA controller 242 to load the packet from LAN memory 236 into FIFO 240, programs FIFO 240 with the direction of data transfer, and programs DMA controller 272 to read the packet out of FIFO 240 and across the VME bus 120 into system memory 116. In particular, the microprocessor 210 first programs the LAN DMA controller 242 with the starting address and length of the packet in LAN memory 236, and programs the

controller to begin transferring data from the LAN  
memory 236 to port A of parity FIFO 240 as soon as the  
FIFO is ready to receive data. Second, microprocessor  
210 programs the VME/FIFO DMA controller 272 with the  
5 destination address in system memory 116 and the length  
of the data packet, and instructs the controller to  
begin transferring data from port B of the FIFO 260  
onto VME bus 120. Finally, the microprocessor 210  
programs FIFO 240 with the direction of the transfer to  
10 take place. The transfer then proceeds entirely under  
the control of DMA controllers 242 and 262, without any  
further involvement by microprocessor 210.

The microprocessor 210 then sends a message to  
host 118 that a packet is available at a specified  
15 system memory address. The microprocessor 210 sends  
such a message by writing a message descriptor to a  
software-emulated command FIFO on the host, which  
copies the message from CPU memory 214 on the NC via  
buffer 284 and into the host's local memory, in  
20 ordinary VME block transfer mode. The host then copies  
the packet from system memory 116 into the host's own  
local memory using ordinary VME transfers.

If the packet received by NC 110a from the network  
is an IP packet, then the microprocessor 210 determines  
25 whether it is (1) an IP packet for the server 100 which  
is not an NFS packet; (2) an IP packet to be routed to

Attorney Docket No.:AUSP7209  
WPI/WSW/AUSP/7209.001

8/24/89-7

a different network; or (3) an NFS packet. If it is an IP packet for the server 100, but not an NFS packet, then the microprocessor 210 causes the packet to be transmitted from the LAN memory 236 to the host 118 in the same manner described above with respect to certain ARP packets.

If the IP packet is not intended for the server 100, but rather is to be routed to a client on a different network, then the packet is copied into the LAN memory associated with the Ethernet to which the destination client is connected. If the destination client is on the Ethernet 122b, which is on the same NC board as the source Ethernet 122a, then the microprocessor 210 causes the packet to be copied from LAN memory 236 into LAN 256 and then causes LAN controller 254 to transmit it over Ethernet 122b. (Of course, if the two LAN data buses 232 and 252 are combined, then copying would be unnecessary; the microprocessor 210 would simply cause the LAN controller 254 to read the packet out of the same locations in LAN memory to which the packet was written by LAN controller 234.)

The copying of a packet from LAN memory 236 to LAN memory 256 takes place similarly to the copying described above from LAN memory to system memory. For transfer sizes of 64 bytes or more, the microprocessor



210 first programs the LAN DMA controller 242 with the  
starting address and length of the packet in LAN memory  
236, and programs the controller to begin transferring  
data from the LAN memory 236 into port A of parity FIFO  
5 240 as soon as the FIFO is ready to receive data.  
Second, microprocessor 210 programs the LAN DMA  
controller 262 with a destination address in LAN memory  
256 and the length of the data packet, and instructs  
that controller to transfer data from parity FIFO 260  
10 into the LAN memory 256. Third, microprocessor 210  
programs the VME/FIFO DMA controller 272 to clock words  
of data out of port B of the FIFO 240, over the data  
bus 274, and into port B of FIFO 260. Finally, the  
microprocessor 210 programs the two FIFOs 240 and 260  
15 with the direction of the transfer to take place. The  
transfer then proceeds entirely under the control of  
DMA controllers 242, 262 and 272, without any further  
involvement by the microprocessor 210. Like the  
copying from LAN memory to system memory, if the  
20 transfer size is smaller than 64 bytes, the  
microprocessor 210 performs the transfer directly,  
without DMA.

When each of the LAN DMA controllers 242 and 262  
complete their work, they so notify microprocessor 210  
25 by a respective interrupt provided through MFP 224.  
When the microprocessor 210 has received both

interrupts, it programs LAN controller 254 to transmit the packet on the Ethernet 122b, in a conventional manner.

Thus, IP routing between the two Ethernets in a  
5 single network controller 110 takes place over data bus 274, generating no traffic over VME bus 120. Nor is the host processor 118 disturbed for such routing, in contrast to the prior art architecture of Fig. 1. Moreover, all but the shortest copying work is  
10 performed by controllers outside microprocessor 210, requiring the involvement of the microprocessor 210, and bus traffic on microprocessor data bus 212, only for the supervisory functions of programming the DMA controllers and the parity FIFOs and instructing them  
15 to begin. The VME/FIFO DMA controller 272 is programmed by loading control registers via microprocessor data bus 212; the LAN DMA controllers 242 and 262 are programmed by loading control registers on the respective controllers via the microprocessor  
20 data bus 212, respective bidirectional buffers 230 and 250, and respective LAN data buses 232 and 252, and the parity FIFOs 240 and 260 are programmed as set forth in the Appendix C.

If the destination workstation of the IP packet to  
25 be routed is on an Ethernet connected to a different one of the network controllers 110, then the packet is

Attorney Docket No.:AUSP7209  
WPI/WSW/AUSP/7209.001

8/24/89-7

copied into the appropriate LAN memory on the NC 110 to which that Ethernet is connected. Such copying is accomplished by first copying the packet into system memory 116, in the manner described above with respect  
5 to certain ARP packets, and then notifying the destination NC that a packet is available. When an NC is so notified, it programs its own parity FIFO and DMA controllers to copy the packet from system memory 116 into the appropriate LAN memory. It is noteworthy that  
10 though this type of IP routing does create VME bus traffic, it still does not involve the host CPU 118.

If the IP packet received over the Ethernet 122a and now stored in LAN memory 236 is an NFS packet intended for the server 100, then the microprocessor  
15 210 performs all necessary protocol preprocessing to extract the NFS message and convert it to the local NFS (LNFS) format. This may well involve the logical concatenation of data extracted from a large number of individual IP packets stored in LAN memory 236,  
20 resulting in a linked list, in CPU memory 214, pointing to the different blocks of data in LAN memory 236 in the correct sequence.

The exact details of the LNFS format are not important for an understanding of the invention, except  
25 to note that it includes commands to maintain a directory of files which are stored on the disks

attached to the storage processors 114, commands for reading and writing data to and from a file on the disks, and various configuration management and diagnostics control messages. The directory maintenance commands which are supported by LNFS include the following messages based on conventional NFS: get attributes of a file (GETATTR); set attributes of a file (SETATTR); look up a file (LOOKUP); create a file (CREATE); remove a file (REMOVE); rename a file (RENAME); create a new linked file (LINK); create a symlink (SYMLINK); remove a directory (RMDIR); and return file system statistics (STATFS). The data transfer commands supported by LNFS include read from a file (READ); write to a file (WRITE); read from a directory (REaddir); and read a link (READLINK). LNFS also supports a buffer release command (RELEASE), for notifying the file controller that an NC is finished using a specified buffer in system memory. It also supports a VOP-derived access command, for determining whether a given type access is legal for specified credential on a specified file.

If the LNFS request includes the writing of file data from the LAN memory 236 to disk, the NC 110a first requests a buffer in system memory 116 to be allocated by the appropriate FC 112. When a pointer to the buffer is returned, microprocessor 210 programs LAN DMA

controller 242, parity FIFO 240 and VME/FIFO DMA controller 272 to transmit the entire block of file data to system memory 116. The only difference between this transfer and the transfer described above for transmitting IP packets and ARP packets to system memory 116 is that these data blocks will typically have portions scattered throughout LAN memory 236. The microprocessor 210 accommodates that situation by programming LAN DMA controller 242 successively for each portion of the data, in accordance with the linked list, after receiving notification that the previous portion is complete. The microprocessor 210 can program the parity FIFO 240 and the VME/FIFO DMA controller 272 once for the entire message, as long as the entire data block is to be placed contiguously in system memory 116. If it is not, then the microprocessor 210 can program the DMA controller 272 for successive blocks in the same manner LAN DMA controller 242.

20 If the network controller 110a receives a message from another processor in server 100, usually from file controller 112, that file data is available in system memory 116 for transmission on one of the Ethernets, for example Ethernet 122a, then the network controller 25 110a copies the file data into LAN memory 236 in a manner similar to the copying of file data in the

opposite direction. In particular, the microprocessor  
210 first programs VME/FIFO DMA controller 272 with the  
starting address and length of the data in system  
memory 116, and programs the controller to begin  
5 transferring data over the VME bus 120 into port B of  
parity FIFO 240 as soon as the FIFO is ready to receive  
data. The microprocessor 210 then programs the LAN DMA  
controller 242 with a destination address in LAN memory  
236 and then length of the file data, and instructs  
10 that controller to transfer data from the parity FIFO  
240 into the LAN memory 236. Third, microprocessor  
210 programs the parity FIFO 240 with the direction of  
the transfer to take place. The transfer then proceeds  
entirely under the control of DMA controllers 242 and  
15 272, without any further involvement by the  
microprocessor 210. Again, if the file data is  
scattered in multiple blocks in system memory 116, the  
microprocessor 210 programs the VME/FIFO DMA controller  
272 with a linked list of the blocks to transfer in the  
20 proper order.

When each of the DMA controllers 242 and 262  
complete their work, they so notify microprocessor 210  
through MFP 224. The microprocessor 210 then performs  
all necessary protocol processing on the LNFS message  
25 in LAN memory 236 in order to prepare the message for  
transmission over the Ethernet 122a in the form of

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Ethernet IP packets. As set forth above, this protocol processing is performed entirely in network controller 110a, without any involvement of the local host 118.

It should be noted that the parity FIFOs are designed to move multiples of 128-byte blocks most efficiently. The data transfer size through port B is always 32-bits wide, and the VME address corresponding to the 32-bit data must be quad-byte aligned. The data transfer size for port A can be either 8 or 16 bits. For bus utilization reasons, it is set to 16 bits when the corresponding local start address is double-byte aligned, and is set at 8 bits otherwise. The TCP/IP checksum is always computed in the 16 bit mode. Therefore, the checksum word requires byte swapping if the local start address is not double-byte aligned.

Accordingly, for transfer from port B to port A of any of the FIFOs 240, 260 or 270, the microprocessor 210 programs the VME/FIFO DMA controller to pad the transfer count to the next 128-byte boundary. The extra 32-bit word transfers do not involve the VME bus, and only the desired number of 32-bit words will be unloaded from port A.

For transfers from port A to port B of the parity FIFO 270, the microprocessor 210 loads port A word-by-word and forces a FIFO full indication when it is finished. The FIFO full indication enables unloading

from port B. The same procedure also takes place for transfers from port A to port B of either of the parity FIFOs 240 or 260, since transfers of fewer than 128 bytes are performed under local microprocessor control rather than under the control of LAN DMA controller 242 or 262. For all of the FIFOs, the VME/FIFO DMA controller is programmed to unload only the desired number of 32-bit words.

FILE CONTROLLER HARDWARE ARCHITECTURE

10 The file controllers (FC) 112 may each be a standard off-the-shelf microprocessor board, such as one manufactured by Motorola Inc. Preferably, however, a more specialized board is used such as that shown in block diagram form in Fig. 4.

15 Fig. 4 shows one of the FCs 112a, and it will be understood that the other FC can be identical. In many aspects it is simply a scaled-down version of the NC 110a shown in Fig. 3, and in some respects it is scaled up. Like the NC 110a, FC 112a comprises a 20MHz 68020  
20 microprocessor 310 connected to a 32-bit microprocessor data bus 312. Also connected to the microprocessor data bus 312 is a 256K byte shared CPU memory 314. The low order 8 bits of the microprocessor data bus 312 are connected through a bidirectional buffer 316 to an  
25 8-bit slow-speed data bus 318. On slow-speed data bus



318 are a 128K byte PROM 320, and a multifunction peripheral (MFP) 324. The functions of the PROM 320 and MFP 324 are the same as those described above with respect to EPROM 220 and MFP 224 on NC 110a. FC 112a  
5 does not include PROM like the PROM 222 on NC 110a, but does include a parallel port 392. The parallel port 392 is mainly for testing and diagnostics.

Like the NC 110a, the FC 112a is connected to the VME bus 120 via a bidirectional buffer 380 and a 32-  
10 bit local data bus 376. A set of control registers 382 are connected to the local data bus 376, and directly addressable across the VME bus 120. The local data bus 376 is also coupled to the microprocessor data bus 312 via a bidirectional buffer 384. This permits the  
15 direct addressability of CPU memory 314 from VME bus 120.

FC 112a also includes a command FIFO 390, which includes an input port coupled to the local data bus 376 and which is directly addressable across the VME  
20 bus 120. The command FIFO 390 also includes an output port connected to the microprocessor data bus 312. The structure, operation and purpose of command FIFO 390 are the same as those described above with respect to command FIFO 290 on NC 110a.

25 The FC 112a omits the LAN data buses 232 and 252 which are present in NC 110a, but instead includes a 4

megabyte 32-bit wide FC memory 396 coupled to the microprocessor data bus 312 via a bidirectional buffer 394. As will be seen, FC memory 396 is used as a cache memory for file control information, separate from the file data information cached in system memory 116.

The file controller embodiment shown in Fig. 4 does not include any DMA controllers, and hence cannot act as a master for transmitting or receiving data in any block transfer mode, over the VME bus 120. Block transfers do occur with the CPU memory 314 and the FC memory 396, however, with the FC 112a acting as an VME bus slave. In such transfers, the remote master addresses the CPU memory 314 or the FC memory 396 directly over the VME bus 120 through the bidirectional buffers 384 and, if appropriate, 394.

#### FILE CONTROLLER OPERATION

The purpose of the FC 112a is basically to provide virtual file system services in response to requests provided in LNFS format by remote processors on the VME bus 120. Most requests will come from a network controller 110, but requests may also come from the local host 118.

The file related commands supported by LNFS are identified above. They are all specified to the FC 112a in terms of logically identified disk data blocks.

For example, the LNFS command for reading data from a file includes a specification of the file from which to read (file system ID (FSID) and file ID (inode)), a byte offset, and a count of the number of bytes to read. The FC 112a converts that identification into physical form, namely disk and sector numbers, in order to satisfy the command.

The FC 112a runs a conventional Fast File System (FFS or UFS), which is based on the Berkeley 4.3 VAX release. This code performs the conversion and also performs all disk data caching and control data caching. However, as previously mentioned, control data caching is performed using the FC memory 396 on FC 112a, whereas disk data caching is performed using the system memory 116 (Fig. 2). Caching this file control information within the FC 112a avoids the VME bus congestion and speed degradation which would result if file control information was cached in system memory 116.

The memory on the FC 112a is directly accessed over the VME bus 120 for three main purposes. First, and by far the most frequent, are accesses to FC memory 396 by an SP 114 to read or write cached file control information. These are accesses requested by FC 112a to write locally modified file control structures through to disk, or to read file control structures

from disk. Second, the FC's CPU memory 314 is accessed directly by other processors for message transmissions from the FC 112a to such other processors. For example, if a data block in system memory is to be transferred to an SP 114 for writing to disk, the FC 112a first assembles a message in its local memory 314 requesting such a transfer. The FC 112a then notifies the SP 114, which copies the message directly from the CPU memory 314 and executes the requested transfer.

10 A third type of direct access to the FC's local memory occurs when an LNFS client reads directory entries. When FC 112a receives an LNFS request to read directory entries, the FC 112a formats the requested directory entries in FC memory 396 and notifies the requestor of their location. The requestor then directly accesses FC memory 396 to read the entries.

The version of the UFS code on FC 112a includes some modifications in order to separate the two caches. In particular, two sets of buffer headers are maintained, one for the FC memory 396 and one for the system memory 116. Additionally, a second set of the system buffer routines (GETBLK(), BRELSE(), BREAD(), BWRITE(), and BREADA()) exist, one for buffer accesses to FC Mem 396 and one for buffer accesses to system memory 116. The UFS code is further modified to call

the appropriate buffer routines for FC memory 396 for  
accesses to file control information, and to call the  
appropriate buffer routines for the system memory 116  
for the caching of disk data. A description of UFS may  
5 be found in chapters 2, 6, 7 and 8 of "Kernel Structure  
and Flow," by Riéken and Webb of .sh consulting (Santa  
Clara, California: 1988), incorporated herein by  
reference.

When a read command is sent to the FC by a  
10 requestor such as a network controller, the FC first  
converts the file, offset and count information into  
disk and sector information. It then locks the system  
memory buffers which contain that information,  
instructing the storage processor 114 to read them from  
15 disk if necessary. When the buffer is ready, the FC  
returns a message to the requestor containing both the  
attributes of the designated file and an array of  
buffer descriptors that identify the locations in  
system memory 116 holding the data.

20 After the requestor has read the data out of the  
buffers, it sends a release request back to the FC.  
The release request is the same message that was  
returned by the FC in response to the read request; the  
FC 112a uses the information contained therein to  
25 determine which buffers to free.

A write command is processed by FC 112a similarly to the read command, but the caller is expected to write to (instead of read from) the locations in system memory 116 identified by the buffer descriptors returned by the FC 112a. Since FC 112a employs write-through caching, when it receives the release command from the requestor, it instructs storage processor 114 to copy the data from system memory 116 onto the appropriate disk sectors before freeing the system memory buffers for possible reallocation.

The READDIR transaction is similar to read and write, but the request is satisfied by the FC 112a directly out of its own FC memory 396 after formatting the requested directory information specifically for this purpose. The FC 112a causes the storage processor read the requested directory information from disk if it is not already locally cached. Also, the specified offset is a "magic cookie" instead of a byte offset, identifying directory entries instead of an absolute byte offset into the file. No file attributes are returned.

The READLINK transaction also returns no file attributes, and since links are always read in their entirety, it does not require any offset or count.

For all of the disk data caching performed through system memory 116, the FC 112a acts as a central

authority for dynamically allocating, deallocating and  
keeping track of buffers. If there are two or more FCs  
112, each has exclusive control over its own assigned  
portion of system memory 116. In all of these  
5 transactions, the requested buffers are locked during  
the period between the initial request and the release  
request. This prevents corruption of the data by other  
clients.

Also in the situation where there are two or more  
10 FCs, each file system on the disks is assigned to a  
particular one of the FCs. FC #0 runs a process called  
FC\_VICE\_PRESIDENT, which maintains a list of which file  
systems are assigned to which FC. When a client  
processor (for example an NC 110) is about to make an  
15 LNFS request designating a particular file system, it  
first sends the fsid in a message to the  
FC\_VICE\_PRESIDENT asking which FC controls the  
specified file system. The FC\_VICE\_PRESIDENT responds,  
and the client processor sends the LNFS request to the  
20 designated FC. The client processor also maintains its  
own list of fsid/FC pairs as it discovers them, so as  
to minimize the number of such requests to the  
FC\_VICE\_PRESIDENT.

STORAGE PROCESSOR HARDWARE ARCHITECTURE

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSE/7209.001

8/24/89-7

In the file server 100, each of the storage processors 114 can interface the VME bus 120 with up to 10 different SCSI buses. Additionally, it can do so at the full usage rate of an enhanced block transfer protocol of 55MB per second.

Fig. 5 is a block diagram of one of the SPs 114a. SP 114b is identical. SP 114a comprises a microprocessor 510, which may be a Motorola 68020 microprocessor operating at 20MHz. The microprocessor 510 is coupled over a 32-bit microprocessor data bus 512 with CPU memory 514, which may include up to 1MB of static RAM. The microprocessor 510 accesses instructions, data and status on its own private bus 512, with no contention from any other source. The microprocessor 510 is the only master of bus 512.

The low order 16 bits of the microprocessor data bus 512 interface with a control bus 516 via a bidirectional buffer 518. The low order 8 bits of the control bus 516 interface with a slow speed bus 520 via another bidirectional buffer 522. The slow speed bus 520 connects to an MFP 524, similar to the MFP 224 in NC 110a (Fig. 3), and with a PROM 526, similar to PROM 220 on NC 110a. The PROM 526 comprises 128K bytes of EPROM which contains the functional code for SP 114a. Due to the width and speed of the PROM 526, the



functional code is copied to CPU memory 514 upon reset for faster execution.

MFP 524, like the MFP 224 on NC 110a, comprises a Motorola 68901 multifunction peripheral device. It provides the functions of a vectored interrupt controller, individually programmable I/O pins, four timers and a UART. The UART functions provide serial communications across an RS 232 bus (not shown in Fig. 5) for debug monitors and diagnostics. Two of the four timing functions may be used as general-purpose timers by the microprocessor 510, either independently or in cascaded fashion. A third timer function provides the refresh clock for a DMA controller described below, and the fourth timer generates the UART clock. Additional information on the MFP 524 can be found in "MC 68901 Multi-Function Peripheral Specification," by Motorola, Inc., which is incorporated herein by reference.

The eight general-purpose I/O bits provided by MFP 524 are configured according to the following table:

Bit Direction Definition

7	input	Power Failure is Imminent - This functions as an early warning.	
5	6	input	SCSI Attention - A composite of the SCSI. Attentions from all 10 SCSI channels.
10	5	input	Channel Operation Done - A composite of the channel done bits from all 13 channels of the DMA controller, described below.
15	4	output	DMA Controller Enable. Enables the DMA Controller to run.
	3	input	VMEbus Interrupt Done - Indicates the completion of a VMEbus Interrupt.
20	2	input	Command Available - Indicates that the SP'S Command Fifo, described below, contains one or more command pointers.
25	1	output	External Interrupts Disable. Disables externally generated interrupts to the microprocessor 510.
	0	output	Command Fifo Enable. Enables operation of the SP'S Command Fifo. Clears the Command Fifo when reset.

30        Commands are provided to the SP 114a from the VME bus 120 via a bidirectional buffer 530, a local data bus 532, and a command FIFO 534. The command FIFO 534 is similar to the command FIFOs 290 and 390 on NC 110a and FC 112a, respectively, and has a depth of 256 32-bit entries. The command FIFO 534 is a write-only register as seen on the VME bus 120, and as a read-only register as seen by microprocessor 510. If the FIFO is full at the beginning of a write from the VME bus, a VME bus error is generated. Pointers are

removed from the command FIFO 534 in the order received, and only by the microprocessor 510. Command available status is provided through I/O bit 4 of the MFP 524, and as long as one or more command pointers  
5 are still within the command FIFO 534, the command available status remains asserted.

As previously mentioned, the SP 114a supports up to 10 SCSI buses or channels 540a-540j. In the typical configuration, buses 540a-540i support up to 3 SCSI  
10 disk drives each, and channel 540j supports other SCSI peripherals such as tape drives, optical disks, and so on. Physically, the SP 114a connects to each of the SCSI buses with an ultra-miniature D sub connector and round shielded cables. Six 50-pin cables provide 300  
15 conductors which carry 18 signals per bus and 12 grounds. The cables attach at the front panel of the SP 114a and to a commutator board at the disk drive array. Standard 50-pin cables connect each SCSI device to the commutator board. Termination resistors are  
20 installed on the SP 114a.

The SP 114a supports synchronous parallel data transfers up to 5MB per second on each of the SCSI buses 540, arbitration, and disconnect/reconnect services. Each SCSI bus 540 is connected to a  
25 respective SCSI adaptor 542, which in the present embodiment is an AIC 6250 controller IC manufactured by

Adaptec Inc., Milpitas, California, operating in the non-multiplexed address bus mode. The AIC 6250 is described in detail in "AIC-6250 Functional Specification," by Adaptec Inc., which is incorporated  
5 herein by reference. The SCSI adaptors 542 each provide the necessary hardware interface and low-level electrical protocol to implement its respective SCSI channel.

The 8-bit data port of each of the SCSI adaptors  
10 542 is connected to port A of a respective one of a set of ten parity FIFOs 544a-544j. The FIFOs 544 are the same as FIFOs 240, 260 and 270 on NC 110a, and are connected and configured to provide parity covered data transfers between the 8-bit data port of the respective  
15 SCSI adaptors 542 and a 36-bit (32-bit plus 4 bits of parity) common data bus 550. The FIFOs 544 provide handshake, status, word assembly/disassembly and speed matching FIFO buffering for this purpose. The FIFOs 544 also generate and check parity for the 32-bit bus,  
20 and for RAID 5 implementations they accumulate and check redundant data and accumulate recovered data.

All of the SCSI adaptors 542 reside at a single location of the address space of the microprocessor 510, as do all of the parity FIFOs 544. The  
25 microprocessor 510 selects individual controllers and FIFOs for access in pairs, by first programming a pair

select register (not shown) to point to the desired pair and then reading from or writing to the control register address of the desired chip in the pair. The microprocessor 510 communicates with the control registers on the SCSI adaptors 542 via the control bus 516 and an additional bidirectional buffer 546, and communicates with the control registers on FIFOs 544 via the control bus 516 and a bidirectional buffer 552. Both the SCSI adaptors 542 and FIFOs 544 employ 8-bit control registers, and register addressing of the FIFOs 544 is arranged such that such registers alias in consecutive byte locations. This allows the microprocessor 510 to write to the registers as a single 32-bit register, thereby reducing instruction overhead.

The parity FIFOs 544 are each configured in their Adaptec 6250 mode. Referring to the Appendix C, the FIFOs 544 are programmed with the following bit settings in the Data Transfer Configuration Register:

20	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
	0	WD Mode	(0)
	1	Parity Chip	(1)
	2	Parity Correct Mode	(0)
	3	8/16 bits CPU & PortA interface	(0)
25	4	Invert Port A address 0	(1)
	5	Invert Port A address 1	(1)

Attorney Docket No.:AUSP7209  
WPI/WSW/AUSP/7209.001

8/24/89-7

6	Checksum Carry Wrap	(0)
7	Reset	(0)

The Data Transfer Control Register is programmed as follows:

5	<u>Bit</u>	<u>Definition</u>	<u>Setting</u> :
	0	Enable PortA Req/Ack	(1)
	1	Enable PortB Req/Ack	(1)
	2	Data Transfer Direction	as desired
	3	CPU parity enable	(0)
10	4	PortA parity enable	(1)
	5	PortB parity enable	(1)
	6	Checksum Enable	(0)
	7	PortA Master	(0)

In addition, bit 4 of the RAM Access Control Register (Long Burst) is programmed for 8-byte bursts.

SCSI adaptors 542 each generate a respective interrupt signal, the status of which are provided to microprocessor 510 as 10 bits of a 16-bit SCSI interrupt register 556. The SCSI interrupt register 556 is connected to the control bus 516. Additionally, a composite SCSI interrupt is provided through the MFP 524 whenever any one of the SCSI adaptors 542 needs servicing.

An additional parity FIFO 554 is also provided in the SP 114a, for message passing. Again referring to

the Appendix C, the parity FIFO 554 is programmed with the following bit settings in the Data Transfer Configuration Register:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
5	0	WD Mode	(0)
	1	Parity Chip	(1)
	2	Parity Correct Mode	(0)
	3	8/16 bits CPU & PortA interface	(1)
	4	Invert Port A address 0	(1)
10	5	Invert Port A address 1	(1)
	6	Checksum Carry Wrap	(0)
	7	Reset	(0)

The Data Transfer Control Register is programmed as follows:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
15	0	Enable PortA Req/Ack	(0)
	1	Enable PortB Req/Ack	(1)
	2	Data Transfer Direction	as desired
	3	CPU parity enable	(0)
20	4	PortA parity enable	(0)
	5	PortB parity enable	(1)
	6	Checksum Enable	(0)
	7	PortA Master	(0)

In addition, bit 4 of the RAM Access Control Register (Long Burst) is programmed for 8-byte bursts.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Port A of FIFO 554 is connected to the 16-bit control bus 516, and port B is connected to the common data bus 550. FIFO 554 provides one means by which the microprocessor 510 can communicate directly with the VME bus 120, as is described in more detail below.

The microprocessor 510 manages data movement using a set of 15 channels, each of which has an unique status which indicates its current state. Channels are implemented using a channel enable register 560 and a channel status register 562, both connected to the control bus 516. The channel enable register 560 is a 16-bit write-only register, whereas the channel status register 562 is a 16-bit read-only register. The two registers reside at the same address to microprocessor 510. The microprocessor 510 enables a particular channel by setting its respective bit in channel enable register 560, and recognizes completion of the specified operation by testing for a "done" bit in the channel status register 562. The microprocessor 510 then resets the enable bit, which causes the respective "done" bit in the channel status register 562 to be cleared.



The channels are defined as follows:

<u>CHANNEL</u>	<u>FUNCTION</u>
5	0:9 These channels control data movement to and from the respective FIFOs 544 via the common data bus 550. When a FIFO is enabled and a request is received from it, the channel becomes ready. Once the channel has been serviced a status of done is generated.
10	11:10 These channels control data movement between a local data buffer 564, described below, and the VME bus 120. When enabled the channel becomes ready. Once the channel has been serviced a status of done is generated.
15	12 When enabled, this channel causes the DRAM in local data buffer 564 to be refreshed based on a clock which is generated by the MFP 524. The refresh consists of a burst of 16 rows. This channel does not generate a status of done.
20	13 The microprocessor's communication FIFO 554 is serviced by this channel. When enable is set and the FIFO 554 asserts a request then the channel becomes ready. This channel generates a status of done.
25	14 Low latency writes from microprocessor 510 onto the VME bus 120 are controlled by this channel. When this channel is enabled data is moved from a special 32 bit register, described below, onto the VME bus 120. This channel generates a done status.
30	15 This is a null channel for which neither a ready status nor done status is generated.
35	

Channels are prioritized to allow servicing of the more critical requests first. Channel priority is assigned in a descending order starting at channel 14. That is, in the event that all channels are requesting service, channel 14 will be the first one served.

The common data bus 550 is coupled via a bidirectional register 570 to a 36-bit junction bus 572. A second bidirectional register 574 connects the junction bus 572 with the local data bus 532. Local data buffer 564, which comprises 1MB of DRAM, with parity, is coupled bidirectionally to the junction bus 572. It is organized to provide 256K 32-bit words with byte parity. The SP 114a operates the DRAMs in page mode to support a very high data rate, which requires bursting of data instead of random single-word accesses. It will be seen that the local data buffer 564 is used to implement a RAID (redundant array of inexpensive disks) algorithm, and is not used for direct reading and writing between the VME bus 120 and a peripheral on one of the SCSI buses 540.

A read-only register 576, containing all zeros, is also connected to the junction bus 572. This register is used mostly for diagnostics, initialization, and clearing of large blocks of data in system memory 116.

The movement of data between the FIFOs 544 and 554, the local data buffer 564, and a remote entity such as the system memory 116 on the VME bus 120, is all controlled by a VME/FIFO DMA controller 580. The VME/FIFO DMA controller 580 is similar to the VME/FIFO DMA controller 272 on network controller 110a (Fig. 3), and is described in the Appendix A. Briefly, it

D

includes a bit slice engine 582 and a dual-port static RAM 584. One port of the dual-port static RAM 584 communicates over the 32-bit microprocessor data bus 512 with microprocessor 510, and the other port communicates over a separate 16-bit bus with the bit slice engine 582. The microprocessor 510 places command parameters in the dual-port RAM 584, and uses the channel enables 560 to signal the VME/FIFO DMA controller 580 to proceed with the command. The VME/FIFO DMA controller is responsible for scanning the channel status and servicing requests, and returning ending status in the dual-port RAM 584. The dual-port RAM 584 is organized as 1K x 32 bits at the 32-bit port and as 2K x 16 bits at the 16-bit port. An example showing the method by which the microprocessor 510 controls the VME/FIFO DMA controller 580 is as follows. First, the microprocessor 510 writes into the dual-port RAM 584 the desired command and associated parameters for the desired channel. For example, the command might be, "copy a block of data from FIFO 544h out into a block of system memory 116 beginning at a specified VME address." Second, the microprocessor sets the channel enable bit in channel enable register 560 for the desired channel.

At the time the channel enable bit is set, the appropriate FIFO may not yet be ready to send data.

Only when the VME/FIFO DMA controller 580 does receive a "ready" status from the channel, will the controller 580 execute the command. In the meantime, the DMA controller 580 is free to execute commands and move data to or from other channels.

When the DMA controller 580 does receive a status of "ready" from the specified channel, the controller fetches the channel command and parameters from the dual-ported RAM 584 and executes. When the command is complete, for example all the requested data has been copied, the DMA controller writes status back into the dual-port RAM 584 and asserts "done" for the channel in channel status register 562. The microprocessor 510 is then interrupted, at which time it reads channel status register 562 to determine which channel interrupted. The microprocessor 510 then clears the channel enable for the appropriate channel and checks the ending channel status in the dual-port RAM 584.

In this way a high-speed data transfer can take place under the control of DMA controller 580, fully in parallel with other activities being performed by microprocessor 510. The data transfer takes place over busses different from microprocessor data bus 512, thereby avoiding any interference with microprocessor instruction fetches.

The SP 114a also includes a high-speed register 590, which is coupled between the microprocessor data bus 512 and the local data bus 532. The high-speed register 590 is used to write a single 32-bit word to an VME bus target with a minimum of overhead. The register is write only as viewed from the microprocessor 510. In order to write a word onto the VME bus 120, the microprocessor 510 first writes the word into the register 590, and the desired VME target address into dual-port RAM 584. When the microprocessor 510 enables the appropriate channel in channel enable register 560, the DMA controller 580 transfers the data from the register 590 into the VME bus address specified in the dual-port RAM 584. The DMA controller 580 then writes the ending status to the dual-port RAM and sets the channel "done" bit in channel status register 562.

This procedure is very efficient for transfer of a single word of data, but becomes inefficient for large blocks of data. Transfers of greater than one word of data, typically for message passing, are usually performed using the FIFO 554.

The SP 114a also includes a series of registers 592, similar to the registers 282 on NC 110a (Fig. 3) and the registers 382 on FC 112a (Fig. 4). The details

of these registers are not important for an understanding of the present invention.

STORAGE PROCESSOR OPERATION

5 The 30 SCSI disk drives supported by each of the SPs 114 are visible to a client processor, for example one of the file controllers 112, either as three large, logical disks or as 30 independent SCSI drives, depending on configuration. When the drives are visible as three logical disks, the SP uses RAID 5 design algorithms to distribute data for each logical drive on nine physical drives to minimize disk arm contention. The tenth drive is left as a spare. The RAID 5 algorithm (redundant array of inexpensive drives, revision 5) is described in "A Case For a Redundant Arrays of Inexpensive Disks (RAID)", by 15 Patterson et al., published at ACM SIGMOD Conference, Chicago, Ill., June 1-3, 1988, incorporated herein by reference.

20 In the RAID 5 design, disk data are divided into stripes. Data stripes are recorded sequentially on eight different disk drives. A ninth parity stripe, the exclusive-or of eight data stripes, is recorded on a ninth drive. If a stripe size is set to 8K bytes, a read of 8K of data involves only one drive. A write of 25 8K of data involves two drives: a data drive and a

parity drive. Since a write requires the reading back  
of old data to generate a new parity stripe, writes are  
also referred to as modify writes. The SP 114a  
supports nine small reads to nine SCSI drives  
5 concurrently. When stripe size is set to 8K, a read of  
64K of data starts all eight SCSI drives, with each  
drive reading one 8K stripe worth of data. The parallel  
operation is transparent to the caller client.

The parity stripes are rotated among the nine  
10 drives in order to avoid drive contention during write  
operations. The parity stripe is used to improve  
availability of data. When one drive is down, the SP  
114a can reconstruct the missing data from a parity  
stripe. In such case, the SP 114a is running in error  
15 recovery mode. When a bad drive is repaired, the SP  
114a can be instructed to restore data on the repaired  
drive while the system is on-line.

When the SP 114a is used to attach thirty  
independent SCSI drives, no parity stripe is created  
20 and the client addresses each drive directly.

The SP 114a processes multiple messages  
(transactions, commands) at one time, up to 200  
messages per second. The SP 114a does not initiate any  
messages after initial system configuration. The  
25 following SP 114a operations are defined:

- 01 No Op
- 02 Send Configuration Data
- 03 Receive Configuration Data
- 05 Read and Write Sectors
- 5 06 Read and Write Cache Pages
- 07 IOCTL Operation
- 08 Dump SP 114a Local Data Buffer
- 09 Start/Stop A SCSI Drive
- 0C Inquiry
- 10 0E Read Message Log Buffer
- 0F Set SP 114a Interrupt

The above transactions are described in detail in the above-identified application entitled MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE. For an understanding of the invention, it will be useful to describe the function and operation of only two of these commands: read and write sectors, and read and write cache pages.

Read and Write Sectors

20 This command, issued usually by an FC 112, causes the SP 114a to transfer data between a specified block of system memory and a specified series of contiguous sectors on the SCSI disks. As previously described in connection with the file controller 112, the particular sectors are identified in physical terms. In

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7



particular, the particular disk sectors are identified by SCSI channel number (0-9), SCSI ID on that channel number (0-2), starting sector address on the specified drive, and a count of the number of sectors to read or write. The SCSI channel number is zero if the SP 114a is operating under RAID 5.

The SP 114a can execute up to 30 messages on the 30 SCSI drives simultaneously. Unlike most of the commands to an SP 114, which are processed by microprocessor 510 as soon as they appear on the command FIFO 534, read and write sectors commands (as well as read and write cache memory commands) are first sorted and queued. Hence, they are not served in the order of arrival.

When a disk access command arrives, the microprocessor 510 determines which disk drive is targeted and inserts the message in a queue for that disk drive sorted by the target sector address. The microprocessor 510 executes commands on all the queues simultaneously, in the order present in the queue for each disk drive. In order to minimize disk arm movements, the microprocessor 510 moves back and forth among queue entries in an elevator fashion.

If no error conditions are detected from the SCSI disk drives, the command is completed normally. When a data check error condition occurs and the SP 114a is

configured for RAID 5, recovery actions using  
redundant data begin automatically. When a drive is  
down while the SP 114a is configured for RAID 5,  
recovery actions similar to data check recovery take  
5 place.

Read/Write Cache Pages

This command is similar to read and write sectors,  
except that multiple VME addresses are provided for  
transferring disk data to and from system memory 116.  
10 Each VME address points to a cache page in system  
memory 116, the size of which is also specified in the  
command. When transferring data from a disk to system  
memory 116, data are scattered to different cache  
pages; when writing data to a disk, data are gathered  
15 from different cache pages in system memory 116.  
Hence, this operation is referred to as a scatter-  
gather function.

The target sectors on the SCSI disks are specified  
in the command in physical terms, in the same manner  
20 that they are specified for the read and write sectors  
command. Termination of the command with or without  
error conditions is the same as for the read and write  
sectors command.

The dual-port RAM 584 in the DMA controller 580  
25 maintains a separate set of commands for each channel

controlled by the bit slice engine 582. As each channel completes its previous operation, the microprocessor 510 writes a new DMA operation into the dual-port RAM 584 for that channel in order to satisfy the next operation on a disk elevator queue.

The commands written to the DMA controller 580 include an operation code and a code indicating whether the operation is to be performed in non-block mode, in standard VME block mode, or in enhanced block mode. The operation codes supported by DMA controller 580 are as follows:

	<u>OP CODE</u>	<u>OPERATION</u>	
	0	NO-OP	
15	1	ZEROES -> BUFFER	Move zeros from zeros register 576 to local data buffer 564.
20	2	ZEROES -> FIFO	Move zeros from zeros register 576 to the currently selected FIFO on common data bus 550.
25	3	ZEROES -> VMEbus	Move zeros from zeros register 576 out onto the VME bus 120. Used for initializing cache buffers in system memory 116.

- 4 VMEbus -> BUFFER Move data from the VME bus 120 to the local data buffer 564. This operation is used during a write, to move target data intended for a down drive into the buffer for participation in redundancy generation. Used only for RAID 5 application.
- 5
- 10
- 5 VMEbus -> FIFO New data to be written from VME bus onto a drive. Since RAID 5 requires redundancy data to be generated from data that is buffered in local data buffer 564, this operation will be used only if the SP 114a is not configured for RAID 5.
- 15
- 20
- 6 VMEbus -> BUFFER & FIFO Target data is moved from VME bus 120 to a SCSI device and is also captured in the local data buffer 564 for participation in redundancy generation. Used only if SP 114a is configured for RAID 5 operation.
- 25
- 30
- 7 BUFFER -> VMEbus This operation is not used.
- 35
- 8 BUFFER -> FIFO Participating data is transferred to create redundant data or recovered data on a disk drive. Used only in RAID 5 applications.
- 40
- 9 FIFO -> VMEbus This operation is used to move target data directly from a disk drive onto the VME bus 120.
- 45

- 5           A       FIFO   -> BUFFER    U s e d   t o   m o v e  
  participating data for  
  recovery and modify  
  operations. Used only in  
  RAID 5 applications.
- 10           B       FIFO   -> VMEbus & BUFFER  
  This operation is used to  
  save target data for  
  participation in data  
  recovery. Used only in  
  RAID 5 applications.

SYSTEM MEMORY

15           Fig. 6 provides a simplified block diagram of the  
                  preferred architecture of one of the system memory  
                  cards 116a. Each of the other system memory cards are  
                  the same. Each memory card 116 operates as a slave on  
                  the enhanced VME bus 120 and therefore requires no on-  
                  board CPU. Rather, a timing control block 610 is  
                  sufficient to provide the necessary slave control  
20           operations. In particular, the timing control block  
                  610, in response to control signals from the control  
                  portion of the enhanced VME bus 120, enables a 32-bit  
                  wide buffer 612 for an appropriate direction transfer  
                  of 32-bit data between the enhanced VME bus 120 and a  
25           multiplexer unit 614. The multiplexer 614 provides a  
                  multiplexing and demultiplexing function, depending on  
                  data transfer direction, for a six megabit by seventy-  
                  two bit word memory array 620. An error correction  
                  code (ECC) generation and testing unit 622 is also  
30           connected to the multiplexer 614 to generate or verify,

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

again depending on transfer direction, eight bits of ECC data. The status of ECC verification is provided back to the timing control block 610.

ENHANCED VME BUS PROTOCOL

5 VME bus 120 is physically the same as an ordinary VME bus, but each of the NCs and SPs include additional circuitry and firmware for transmitting data using an enhanced VME block transfer protocol. The enhanced protocol is described in detail in the above-identified  
10 application entitled ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER, and summarized in the Appendix B hereto. Typically transfers of LNFS file data between NCs and system memory, or between SPs and system memory, and  
15 transfers of packets being routed from one NC to another through system memory, are the only types of transfers that use the enhanced protocol in server 100. All other data transfers on VME bus 120 use either conventional VME block transfer protocols or ordinary  
20 non-block transfer protocols.

MESSAGE PASSING

As is evident from the above description, the different processors in the server 100 communicate with each other via certain types of messages. In software,

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

these messages are all handled by the messaging kernel, described in detail in the MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE application cited above. In hardware, they are implemented as follows.

5           Each of the NCs 110, each of the FCs 112, and each of the SPs 114 includes a command or communication FIFO such as 290 on NC 110a. The host 118 also includes a command FIFO, but since the host is an unmodified purchased processor board, the FIFO is emulated in  
10 software. The write port of the command FIFO in each of the processors is directly addressable from any of the other processors over VME bus 120.

          Similarly, each of the processors except SPs 114 also includes shared memory such as CPU memory 214 on  
15 NC 110a. This shared memory is also directly addressable by any of the other processors in the server 100.

          If one processor, for example network controller 110a, is to send a message or command to a second  
20 processor, for example file controller 112a, then it does so as follows. First, it forms the message in its own shared memory (e.g., in CPU memory 214 on NC 110a). Second, the microprocessor in the sending processor directly writes a message descriptor into the command  
25 FIFO in the receiving processor. For a command being sent from network controller 110a to file controller

112a, the microprocessor 210 would perform the write via buffer 284 on NC 110a, VME bus 120, and buffer 384 on file controller 112a.

The command descriptor is a single 32-bit word containing in its high order 30 bits a VME address indicating the start of a quad-aligned message in the sender's shared memory. The low order two bits indicate the message type as follows:

	<u>Type</u>	<u>Description</u>
10	0	Pointer to a new message being sent
	1	Pointer to a reply message
	2	Pointer to message to be forwarded
	3	Pointer to message to be freed; also message acknowledgment

15 All messages are 128-bytes long.

When the receiving processor reaches the command descriptor on its command FIFO, it directly accesses the sender's shared memory and copies it into the receiver's own local memory. For a command issued from network controller 110a to file controller 112a, this would be an ordinary VME block or non-block mode transfer from NC CPU memory 214, via buffer 284, VME bus 120 and buffer 384, into FC CPU memory 314. The FC microprocessor 310 directly accesses NC CPU memory 214 for this purpose over the VME bus 120.

When the receiving processor has received the command and has completed its work, it sends a reply

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7



message back to the sending processor. The reply message may be no more than the original command message unaltered, or it may be a modified version of that message or a completely new message. If the reply message is not identical to the original command message, then the receiving processor directly accesses the original sender's shared memory to modify the original command message or overwrite it completely. For replies from the FC 112a to the NC 110a, this involves an ordinary VME block or non-block mode transfer from the FC 112a, via buffer 384, VME bus 120, buffer 284 and into NC CPU memory 214. Again, the FC microprocessor 310 directly accesses NC CPU memory 214 for this purpose over the VME bus 120.

Whether or not the original command message has been changed, the receiving processor then writes a reply message descriptor directly into the original sender's command FIFO. The reply message descriptor contains the same VME address as the original command message descriptor, and the low order two bits of the word are modified to indicate that this is a reply message. For replies from the FC 112a to the NC 110a, the message descriptor write is accomplished by microprocessor 310 directly accessing command FIFO 290 via buffer 384, VME bus 120 and buffer 280 on the NC. Once this is done, the receiving processor can free the

buffer in its local memory containing the copy of the command message.

When the original sending processor reaches the reply message descriptor on its command FIFO, it wakes up the process that originally sent the message and permits it to continue. After examining the reply message, the original sending processor can free the original command message buffer in its own local shared memory.

As mentioned above, network controller 110a uses the buffer 284 data path in order to write message descriptors onto the VME bus 120, and uses VME/FIFO DMA controller 272 together with parity FIFO 270 in order to copy messages from the VME bus 120 into CPU memory 214. Other processors read from CPU memory 214 using the buffer 284 data path.

File controller 112a writes message descriptors onto the VME bus 120 using the buffer 384 data path, and copies messages from other processors' shared memory via the same data path. Both take place under the control of microprocessor 310. Other processors copy messages from CPU memory 314 also via the buffer 384 data path.

Storage processor 114a writes message descriptors onto the VME bus using high-speed register 590 in the manner described above, and copies messages from other

processors using DMA controller 580 and FIFO 554. The  
SP 114a has no shared memory, however, so it uses a  
buffer in system memory 116 to emulate that function.  
That is, before it writes a message descriptor into  
5 another processor's command FIFO, the SP 114a first  
copies the message into its own previously allocated  
buffer in system memory 116 using DMA controller 580  
and FIFO 554. The VME address included in the message  
descriptor then reflects the VME address of the message  
10 in system memory 116.

In the host 118, the command FIFO and shared  
memory are both emulated in software.

The invention has been described with respect to  
particular embodiments thereof, and it will be  
15 understood that numerous modifications and variations  
are possible within the scope of the invention.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

APPENDIX A

VME/FIFO DMA Controller

In storage processor 114a, DMA controller 580 manages the data path under the direction of the microprocessor 510. The DMA controller 580 is a microcoded 16-bit bit-slice implementation executing pipelined instructions at a rate of one each 62.5ns. It is responsible for scanning the channel status 562 and servicing request with parameters stored in the dual-ported ram 584 by the microprocessor 510. Ending status is returned in the ram 584 and interrupts are generated for the microprocessor 510.

Control Store. The control store contains the microcoded instructions which control the DMA controller 580. The control store consists of 6 1K x 8 proms configured to yield a 1K x 48 bit microword. Locations within the control store are addressed by the sequencer and data is presented at the input of the pipeline registers.

Sequencer. The sequencer controls program flow by generating control store addresses based upon pipeline data and various status bits. The control store address consists of 10 bits. Bits 8:0 of the control store address derive from a multiplexer having as its inputs either an ALU output or the output of an incrementer. The incrementer can be preloaded with

pipeline register bits 8:0, or it can be incremented as a result of a test condition. The 1K address range is divided into two pages by a latched flag such that the microprogram can execute from either page. Branches, however remain within the selected page. Conditional sequencing is performed by having the test condition increment the pipeline provided address. A false condition allows execution from the pipeline address while a true condition causes execution from the address + 1. The alu output is selected as an address source in order to directly vector to a routine or in order to return to a calling routine. Note that when calling a subroutine the calling routine must reside within the same page as the subroutine or the wrong page will be selected on the return.

ALU. The alu comprises a single IDT49C402A integrated circuit. It is 16 bits in width and most closely resembles four 2901s with 64 registers. The alu is used primarily for incrementing, decrementing, addition and bit manipulation. All necessary control signals originate in the control store. The IDT HIGH PERFORMANCE CMOS 1988 DATA BOOK, incorporated by reference herein, contains additional information about the alu.

Microword. The 48 bit microword comprises several fields which control various functions of the

DMA controller 580. The format of the microword is defined below along with mnemonics and a description of each function.

- 5 AI<8:0> 47:39 (Alu Instruction bits 8:0) The AI bits provide the instruction for the 49C402A alu. Refer to the IDT data book for a complete definition of the alu instructions. Note that the I9 signal input of the 49C402A is always low.
- 10 CIN 38 (Carry INput) This bit forces the carry input to the alu.
- 15 RA<5:0> 37:32 (Register A address bits 5:0) These bits select one of 64 registers as the "A" operand for the alu. These bits also provide literal bits 15:10 for the alu bus.
- 20 RB<5:0> 31:26 (Register B address bits 5:0) These bits select one of 64 registers as the "B" operand for the alu. These bits also provide literal bits 9:4 for the alu bus.
- 25 LFD 25 (Latched Flag Data) When set this bit causes the selected latched flag to be set. When reset this bit causes the selected latched flag to be cleared. This bits also functions as literal bit 3 for the alu bus.
- 30 LFS<2:0> 24:22 (Latched Flag Select bits 2:0) The meaning of these bits is dependent upon the selected source for the alu bus. In the event that the literal field is selected as the bus source then LFS<2:0> function as literal bits <2:0> otherwise the bits are used to select one of the latched flags.
- 35

LFS<2:0> SELECTED FLAG

- 0 This value selects a null flag.
- 5 1 When set this bit enables the buffer clock. When reset this bit disables the buffer clock.
- 10 2 When this bit is cleared VME bus transfers, buffer operations and RAS are all disabled.
- 15 3 NOT USED
- 4 When set this bit enables VME bus transfers.
- 20 5 When set this bit enables buffer operations.
- 25 6 When set this bit asserts the row address strobe to the dram buffer.
- 7 When set this bit selects page 0 of the control store.
- 30 SRC<1,0> 20,21 (alu bus SouRCe select bits 1,0) These bits select the data source to be enabled onto the alu bus.

SRC<1,0> Selected Source

- 35 0 alu
- 1 dual ported ram
- 2 literal
- 3 reserved-not defined
- 40 PF<2:0> 19:17 (Pulsed Flag select bits 2:0) These bits select a flag/signal to be pulsed.

PF<2:0> Flag

- 0 null
- 5 1 SGL\_CLK  
generates a single transition  
of buffer clock.
- 10 2 SET\_VB  
forces vme and buffer enable  
to be set.
- 15 3 CL\_PERR  
clears buffer parity error  
status.
- 20 4 SET\_DN  
set channel done status for  
the currently selected  
channel.
- 25 5 INC\_ADR  
increment dual ported ram  
address.
- 6:7 RESERVED - NOT DEFINED

DEST<3:0> 16:13 (DESTINATION select bits 3:0) These bits select one of 10 destinations to be loaded from the alu bus.

DEST<3:0> Destination

- 0 null
- 35 1 WR\_RAM  
causes the data on the alu bus  
to be written to the dual  
ported ram.  
D<15:0> -> ram<15:0>
- 40 2 WR\_BADD  
loads the data from the alu  
bus into the dram address  
counters.  
D<14:7> -> mux addr<8:0>
- 45



-91-

3      WR\_VADL  
loads the data from the alu  
bus into the least significant  
2 bytes of the VME address  
register.  
D<15:2> -> VME addr<15:2>  
D1      -> ENB\_ENH  
D0      -> ENB\_BLK

5

10     4      WR\_VADH  
loads the most significant 2  
bytes of the VME address  
register.  
D<15:0> -> VME addr<31:16>

15

5      WR\_RADD  
loads the dual ported ram  
address counters.  
D<10:0> -> ram addr <10:0>

20

6      WR\_WCNT  
loads the word counters.  
D15     -> count enable\*  
D<14:8> -> count <6:0>

25

7      WR\_CO  
loads the co-channel select  
register.  
D<7:4> -> CO<3:0>

30

8      WR\_NXT  
loads the next-channel select  
register.  
D<3:0> -> NEXT<3:0>

35

9      WR\_CUR  
loads the current-channel  
select register.  
D<3:0> -> CURR <3:0>

40

10:14   RESERVED - NOT DEFINED

15     JUMP  
causes the control store  
sequencer to select the alu  
data bus.  
D<8:0> -> CS\_A<8:0>

45

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSE/7209.001

8/24/89-7

TEST<3:0> 12:9 (TEST condition select bits 3:0) Select one of 16 inputs to the test multiplexor to be used as the carry input to the incrementer.

5

TEST<3:0> Condition

	0	FALSE	-always false
	1	TRUE	-always true
10	2	ALU_COUT	-carry output of alu
	3	ALU_EQ	-equals output of alu
15	4	ALU_OVR	-alu overflow
	5	ALU_NEG	-alu negative
20	6	XFR_DONE	-transfer complete
	7	PAR_ERR	-buffer parity error
	8	TIMOUT	-bus operation timeout
25	9	ANY_ERR	-any error status
	14:10	RESERVED	-NOT DEFINED
30	15	CH_RDY	-next channel ready

NEXT\_A<8:0> 8:0 (NEXT Address bits 8:0) Selects an instructions from the current page of the control store for execution.

Dual Ported Ram. The dual ported ram is the medium by which command, parameters and status are communicated between the DMA controller 580 and the microprocessor 510. The ram is organized as 1K x 32 at the master port and as 2K x 16 at the DMA port. The ram may be both written and read at either port.

The ram is addressed by the DMA controller 580 by loading an 11 bit address into the address counters.

Data is then read into bidirectional registers and the address counter is incremented to allow read of the next location.

5 Writing the ram is accomplished by loading data from the processor into the registers after loading the ram address. Successive writes may be performed on every other processor cycle.

The ram contains current block pointers, ending status, high speed bus address and parameter blocks.

10 The following is the format of the ram:

OFFSET	31	0
0	CURR POINTER 0	STATUS 0
5	4	INITIAL POINTER 0
10	58	CURR POINTER B
	5C	INITIAL POINTER B
15	60	not used
	64	not used
20	68	CURR POINTER D
	6C	INITIAL POINTER D
	70	not used
	74	STATUS E
25	74	HIGH SPEED BUS ADDRESS 31:2 0 0
	78	PARAMETER BLOCK 0
30	??	PARAMETER BLOCK n

The Initial Pointer is a 32 bit value which points  
35 the first command block of a chain. The current pointer  
is a sixteen bit value used by the DMA controller 580  
to point to the current command block. The current  
command block pointer should be initialized to 0x0000  
by the microprocessor 510 before enabling the channel.  
40 Upon detecting a value of 0x0000 in the current block  
pointer the DMA controller 580 will copy the lower 16  
bits from the initial pointer to the current pointer.  
Once the DMA controller 580 has completed the specified

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

operations for the parameter block the current pointer will be updated to point to the next block. In the event that no further parameter blocks are available the pointer will be set to 0x0000.

5 The status byte indicates the ending status for the last channel operation performed. The following status bytes are defined:

	STATUS	MEANING
	0	NO ERRORS
10	1	ILLEGAL OP CODE
	2	BUS OPERATION TIMEOUT
	3	BUS OPERATION ERROR
	4	DATA PATH PARITY ERROR

The format of the parameter block is:

15	OFFSET	31	0
	0	FORWARD LINK	
	4	NOT USED	WORD COUNT
20	8	VME ADDRESS 31:2, ENH, BLK	
	C	TERM 0	OP 0   BUF ADDR 0
25		.	.
		.	.
30	C+(4Xn)	TERM n	OP n   BUF ADDR n

FORWARD LINK - The forward link points to the first word of the next parameter block for execution.  
35 It allows several parameter blocks to be initialized

and chained to create a sequence of operations for execution. The forward pointer has the following format:

A31:A2,0,0

5 The format dictates that the parameter block must start on a quad byte boundary. A pointer of 0x00000000 is a special case which indicates no forward link exists.

WORD COUNT - The word count specifies the number of quad byte words that are to be transferred to or from each buffer address or to/from the VME address. A word count of 64K words may be specified by initializing the word count with the value of 0. The word count has the following format:

|D15|D14|D13|D12|D11|D10|D9|D8|D7|D6|D5|D4|D3|D2|D1|D0|

15 The word count is updated by the DMA controller 580 at the completion of a transfer to/from the last specified buffer address. Word count is not updated after transferring to/from each buffer address and is therefore not an accurate indicator of the total data moved to/from the buffer. Word count represents the amount of data transferred to the VME bus or one of the FIFOs 544 or 554.

20 VME ADDRESS - The VME address specifies the starting address for data transfers. Thirty bits allows the address to start at any quad byte boundary.

ENH - This bit when set selects the enhanced block transfer protocol described in the above-cited ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER application, to be used during the VME bus transfer. Enhanced protocol will be disabled automatically when performing any transfer to or from 24 bit or 16 bit address space, when the starting address is not 8 byte aligned or when the word count is not even.

BLK - This bit when set selects the conventional VME block mode protocol to be used during the VME bus transfer. Block mode will be disabled automatically when performing any transfer to or from 16 bit address space.

BUF ADDR - The buffer address specifies the starting buffer address for the adjacent operation. Only 16 bits are available for a 1M byte buffer and as a result the starting address always falls on a 16 byte boundary. The programmer must ensure that the starting address is on a modulo 128 byte boundary. The buffer address is updated by the DMA controller 580 after completion of each data burst.

|A19|A18|A17|A16|A15|A14|A13|A12|A11|A10|A9|A8|A7|A6|A5|A4|

TERM - The last buffer address and operation within a parameter block is identified by the terminal bit. The DMA controller 580 continues to fetch buffer

addresses and operations to perform until this bit is encountered. Once the last operation within the parameter block is executed the word counter is updated and if not equal to zero the series of operations is repeated. Once the word counter reaches zero the forward link pointer is used to access the next parameter block.

5

|0|0|0|0|0|0|0|0|T|

OP - Operations are specified by the op code. The op code byte has the following format:

10

|0|0|0|0|OP3|OP2|OP1|OP0|

The op codes are listed below ("FIFO" refers to any of the FIFOs 544 or 554):



	<u>OP_CODE</u>	<u>OPERATION</u>
	0	NO-OP
	1	ZEROES -> BUFFER
	2	ZEROES -> FIFO
5	3	ZEROES -> VMEbus
	4	VMEbus -> BUFFER
	5	VMEbus -> FIFO
	6	VMEbus -> BUFFER & FIFO
	7	BUFFER -> VMEbus
10	8	BUFFER -> FIFO
	9	FIFO -> VMEbus
	A	FIFO -> BUFFER
	B	FIFO -> VMEbus & BUFFER
	C	RESERVED
15	D	RESERVED
	E	RESERVED
	F	RESERVED

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

APPENDIX B

Enhanced VME Block Transfer Protocol

The enhanced VME block transfer protocol is a VMEbus compatible pseudo-synchronous fast transfer handshake protocol for use on a VME backplane bus having a master functional module and a slave functional module logically interconnected by a data transfer bus. The data transfer bus includes a data strobe signal line and a data transfer acknowledge signal line. To accomplish the handshake, the master transmits a data strobe signal of a given duration on the data strobe line. The master then awaits the reception of a data transfer acknowledge signal from the slave module on the data transfer acknowledge signal line. The slave then responds by transmitting data transfer acknowledge signal of a given duration on the data transfer acknowledge signal line.

Consistent with the pseudo-synchronous nature of the handshake protocol, the data to be transferred is referenced to only one signal depending upon whether the transfer operation is a READ or WRITE operation. In transferring data from the master functional unit to the slave, the master broadcasts the data to be transferred. The master asserts a data strobe signal and the slave, in response to the data strobe signal, captures the data broadcast by the master. Similarly,

in transferring data from the slave to the master, the slave broadcasts the data to be transferred to the master unit. The slave then asserts a data transfer acknowledge signal and the master, in response to the data transfer acknowledge signal, captures the data broadcast by the slave.

The fast transfer protocol, while not essential to the present invention, facilitates the rapid transfer of large amounts of data across a VME backplane bus by substantially increasing the data transfer rate. These data rates are achieved by using a handshake wherein the data strobe and data transfer acknowledge signals are functionally decoupled and by specifying high current drivers for all data and control lines.

The enhanced pseudo-synchronous method of data transfer (hereinafter referred to as "fast transfer mode") is implemented so as to comply and be compatible with the IEEE VME backplane bus standard. The protocol utilizes user-defined address modifiers, defined in the VMEbus standard, to indicate use of the fast transfer mode. Conventional VMEbus functional units, capable only of implementing standard VMEbus protocols, will ignore transfers made using the fast transfer mode and, as a result, are fully compatible with functional units capable of implementing the fast transfer mode.

The fast transfer mode reduces the number of bus propagations required to accomplish a handshake from four propagations, as required under conventional VMEbus protocols, to only two bus propagations. Likewise, the number of bus propagations required to effect a BLOCK READ or BLOCK WRITE data transfer is reduced. Consequently, by reducing the propagations across the VMEbus to accomplish handshaking and data transfer functions, the transfer rate is materially increased.

The enhanced protocol is described in detail in the above-cited ENHANCED VMEBUS PROTOCOL application, and will only be summarized here. Familiarity with the conventional VME bus standards is assumed.

In the fast transfer mode handshake protocol, only two bus propagations are used to accomplish a handshake, rather than four as required by the conventional protocol. At the initiation of a data transfer cycle, the master will assert and deassert DS0\* in the form of a pulse of a given duration. The deassertion of DS0\* is accomplished without regard as to whether a response has been received from the slave. The master then waits for an acknowledgement from the slave. Subsequent pulsing of DS0\* cannot occur until a responsive DTACK\* signal is received from the slave. Upon receiving the slave's assertion of DTACK\*, the

master can then immediately reassert data strobe, if so  
desired. The fast transfer mode protocol does not  
require the master to wait for the deassertion of  
DTACK\* by the slave as a condition precedent to  
5 subsequent assertions of DSO\*. In the fast transfer  
mode, only the leading edge (i.e., the assertion) of a  
signal is significant. Thus, the deassertion of either  
DSO\* or DTACK\* is completely irrelevant for completion  
of a handshake. The fast transfer protocol does not  
10 employ the DSI\* line for data strobe purposes at all.

The fast transfer mode protocol may be  
characterized as pseudo-synchronous as it includes both  
synchronous and asynchronous aspects. The fast  
transfer mode protocol is synchronous in character due  
15 to the fact that DSO\* is asserted and deasserted  
without regard to a response from the slave. The  
asynchronous aspect of the fast transfer mode protocol  
is attributable to the fact that the master may not  
subsequently assert DSO\* until a response to the prior  
20 strobe is received from the slave. Consequently,  
because the protocol includes both synchronous and  
asynchronous components, it is most accurately  
classified as "pseudo-synchronous."

The transfer of data during a BLOCK WRITE cycle in  
25 the fast transfer protocol is referenced only to DSO\*.  
The master first broadcasts valid data to the slave,

and then asserts DSO to the slave. The slave is given a predetermined period of time after the assertion of DSO\* in which to capture the data. Hence, slave modules must be prepared to capture data at any time, as DTACK\* is not referenced during the transfer cycle.

Similarly, the transfer of data during a BLOCK READ cycle in the fast transfer protocol is referenced only to DTACK\*. The master first asserts DSO\*. The slave then broadcasts data to the master and then asserts DTACK\*. The master is given a predetermined period of time after the assertion of DTACK in which to capture the data. Hence, master modules must be prepared to capture data at any time as DSO is not referenced during the transfer cycle.

Fig. 7, parts A through C, is a flowchart illustrating the operations involved in accomplishing the fast transfer protocol BLOCK WRITE cycle. To initiate a BLOCK WRITE cycle, the master broadcasts the memory address of the data to be transferred and the address modifier across the DTB bus. The master also drives interrupt acknowledge signal (IACK\*) high and the LWORD\* signal low 701. A special address modifier, for example "1F," broadcast by the master indicates to the slave module that the fast transfer protocol will be used to accomplish the BLOCK WRITE.

The starting memory address of the data to be

transferred should reside on a 64-bit boundary and the size of block of data to be transferred should be a multiple of 64 bits. In order to remain in compliance with the VMEbus standard, the block must not cross a 256 byte boundary without performing a new address cycle.

5 The slave modules connected to the DTB receive the address and the address modifier broadcast by the master across the bus and receive LWORD\* low and IACK\* high 703. Shortly after broadcasting the address and address modifier 701, the master drives the AS\* signal low 705. The slave modules receive the AS\* low signal 707. Each slave individually determines whether it will participate in the data transfer by determining whether the broadcasted address is valid for the slave in question 709. If the address is not valid, the data transfer does not involve that particular slave and it ignores the remainder of the data transfer cycle.

10 The master drives WRITE\* low to indicate that the transfer cycle about to occur is a WRITE operation 711. The slave receives the WRITE\* low signal 713 and, knowing that the data transfer operation is a WRITE operation, awaits receipt of a high to low transition on the DS0\* signal line 715. The master will wait until both DTACK\* and BERR\* are high 718, which

indicates that the previous slave is no longer driving the DTB.

5 The master proceeds to place the first segment of the data to be transferred on data lines D00 through D31, 719. After placing data on D00 through D31, the master drives DS0\* low 721 and, after a predetermined interval, drives DS0\* high 723.

10 In response to the transition of DS0\* from high to low, respectively 721 and 723, the slave latches the data being transmitted by the master over data lines D00 through D31, 725. The master places the next segment of the data to be transferred on data lines D00 through D31, 727, and awaits receipt of a DTACK\* signal in the form of a high to low transition signal, 729 in Fig. 7B.

15 Referring to Fig. 7B, the slave then drives DTACK\* low, 731, and, after a predetermined period of time, drives DTACK high, 733. The data latched by the slave, 725, is written to a device, which has been selected to store the data 735. The slave also increments the device address 735. The slave then waits for another transition of DS0\* from high to low 737.

20 To commence the transfer of the next segment of the block of data to be transferred, the master drives DS0\* low 739 and, after a predetermined period of time, drives DS0\* high 741. In response to the



transition of DS0\* from high to low, respectively 739  
and 741, the slave latches the data being broadcast by  
the master over data lines D00 through D31, 743. The  
master places the next segment of the data to be  
5 transferred on data lines D00 through D31, 745, and  
awaits receipt of a DTACK\* signal in the form of a high  
to low transition, 747.

The slave then drives DTACK\* low, 749, and, after  
a predetermined period of time, drives DTACK\* high,  
10 751. The data latched by the slave, 743, is written to  
the device selected to store the data and the device  
address is incremented 753. The slave waits for  
another transition of DS0\* from high to low 737.

The transfer of data will continue in the above-  
15 described manner until all of the data has been  
transferred from the master to the slave. After all of  
the data has been transferred, the master will release  
the address lines, address modifier lines, data lines,  
IACK\* line, LWORD\* line and DS0\* line, 755. The  
20 master will then wait for receipt of a DTACK\* high to  
low transition 757. The slave will drive DTACK\* low,  
759 and, after a predetermined period of time, drive  
DTACK\* high 761. In response to the receipt of the  
DTACK\* high to low transition, the master will drive  
25 AS\* high 763 and then release the AS\* line 765.

Fig. 8, parts A through C, is a flowchart illustrating the operations involved in accomplishing the fast transfer protocol BLOCK READ cycle. To initiate a BLOCK READ cycle, the master broadcasts the memory address of the data to be transferred and the address modifier across the DTB bus 801. The master drives the LWORD\* signal low and the IACK\* signal high 801. As noted previously, a special address modifier indicates to the slave module that the fast transfer protocol will be used to accomplish the BLOCK READ.

The slave modules connected to the DTB receive the address and the address modifier broadcast by the master across the bus and receive LWORD\* low and IACK\* high 803. Shortly after broadcasting the address and address modifier 801, the master drives the AS\* signal low 805. The slave modules receive the AS\* low signal 807. Each slave individually determines whether it will participate in the data transfer by determining whether the broadcasted address is valid for the slave in question 809. If the address is not valid, the data transfer does not involve that particular slave and it ignores the remainder of the data transfer cycle.

The master drives WRITE\* high to indicate that the transfer cycle about to occur is a READ operation 811. The slave receives the WRITE\* high signal 813 and, knowing that the data transfer operation is a READ

operation, places the first segment of the data to be transferred on data lines D00 through D31 819. The master will wait until both DTACK\* and BERR\* are high 818, which indicates that the previous slave is no longer driving the DTB.

5 The master then drives DS0\* low 821 and, after a predetermined interval, drives DS0\* high 823. The master then awaits a high to low transition on the DTACK\* signal line 824. As shown in Fig. 8B, the slave then drives the DTACK\* signal low 825 and, after a predetermined period of time, drives the DTACK\* signal high 827.

10 In response to the transition of DTACK\* from high to low, respectively 825 and 827, the master latches the data being transmitted by the slave over data lines D00 through D31, 831. The data latched by the master, 831, is written to a device, which has been selected to store the data the device address is incremented 833.

15 The slave places the next segment of the data to be transferred on data lines D00 through D31, 829, and then waits for another transition of DS0\* from high to low 835.

20 To commence the transfer of the next segment of the block of data to be transferred, the master drives DS0\* low 839 and, after a predetermined period of

time, drives DS0\* high 841. The master then waits for the DTACK\* line to transition from high to low, 843.

The slave drives DTACK\* low, 845, and, after a predetermined period of time, drives DTACK\* high, 847.

5 In response to the transition of DTACK\* from high to low, respectively 839 and 841, the master latches the data being transmitted by the slave over data lines D00 through D31, 845. The data latched by the master, 845, is written to the device selected to store the data, 10 851 in Fig. 8C, and the device address is incremented. The slave places the next segment of the data to be transferred on data lines D00 through D31, 849.

The transfer of data will continue in the above-described manner until all of the data to be 15 transferred from the slave to the master has been written into the device selected to store the data. After all of the data to be transferred has been written into the storage device, the master will release the address lines, address modifier lines, data 20 lines, the IACK\* line, the LWORD line and DS0\* line 852. The master will then wait for receipt of a DTACK\* high to low transition 853. The slave will drive DTACK\* low 855 and, after a predetermined period of time, drive DTACK\* high 857. In response to the 25 receipt of the DTACK\* high to low transition, the

master will drive AS\* high 859 and release the AS\* line 861.

To implement the fast transfer protocol, a conventional 64 mA tri-state driver is substituted for the 48 mA open collector driver conventionally used in VME slave modules to drive DTACK\*. Similarly, the conventional VMEbus data drivers are replaced with 64 mA tri-state drivers in SO-type packages. The latter modification reduces the ground lead inductance of the actual driver package itself and, thus, reduces "ground bounce" effects which contribute to skew between data, DS0\* and DTACK\*. In addition, signal return inductance along the bus backplane is reduced by using a connector system having a greater number of ground pins so as to minimize signal return and mated-pair pin inductance. One such connector system is the "High Density Plus" connector, Model No. 420-8015-000, manufactured by Teradyne Corporation.

APPENDIX C

Parity FIFO

5 The parity FIFOs 240, 260 and 270 (on the network  
controllers 110), and 544 and 554 (on storage  
processors 114) are each implemented as an ASIC. All  
the parity FIFOs are identical, and are configured on  
power-up or during normal operation for the particular  
function desired. The parity FIFO is designed to allow  
speed matching between buses of different speed, and  
10 to perform the parity generation and correction for  
the parallel SCSI drives.

The FIFO comprises two bidirectional data ports,  
Port A and Port B, with 36 x 64 bits of RAM buffer  
between them. Port A is 8 bits wide and Port B is 32  
15 bits wide. The RAM buffer is divided into two parts,  
each 36 x 32 bits, designated RAM X and RAM Y. The two  
ports access different halves of the buffer alternating  
to the other half when available. When the chip is  
configured as a parallel parity chip (e.g. one of the  
20 FIFOs 544 on SP 114a), all accesses on Port B are  
monitored and parity is accumulated in RAM X and RAM Y  
alternately.

The chip also has a CPU interface, which may be 8  
or 16 bits wide. In 16 bit mode the Port A pins are  
25 used as the most significant data bits of the CPU

interface and are only actually used when reading or writing to the Fifo Data Register inside the chip.

5 A REQ, ACK handshake is used for data transfer on both Ports A and B. The chip may be configured as either a master or a slave on Port A in the sense that, in master mode the Port A ACK / RDY output signifies that the chip is ready to transfer data on Port A, and the Port A REQ input specifies that the slave is responding. In slave mode, however, the Port A REQ 10 input specifies that the master requires a data transfer, and the chip responds with Port A ACK / RDY when data is available. The chip is a master on Port B since it raises Port B REQ and waits for Port B ACK to indicate completion of the data transfer.

15 SIGNAL DESCRIPTIONS

Port A 0-7, P

Port A is the 8 bit data port. Port A P, if used, is the odd parity bit for this port.

A Req, A Ack/Rdy

20 These two signals are used in the data transfer mode to control the handshake of data on Port A.

uP Data 0-7, uP Data P, uPAdd 0-2, CS

These signals are used by a microprocessor to address the programmable registers within the chip. The odd parity signal uP Data P is only checked when data is written to the Fifo Data or Checksum Registers and microprocessor parity is enabled.

Clk

The clock input is used to generate some of the chip timing. It is expected to be in the 10-20 Mhz range.

Read En, Write En

During microprocessor accesses, while CS is true, these signals determine the direction of the microprocessor accesses. During data transfers in the WD mode these signals are data strobes used in conjunction with Port A Ack.

Port B 00-07, 10-17, 20-27, 30-37, 0P-3P

Port B is a 32 bit data port. There is one odd parity bit for each byte. Port B 0P is the parity of bits 00-07, Port B 1P is the parity of bits 10-17, Port B 2P is the parity of bits 20-27, and Port B 3P is the parity of bits 30-37.



**B Select, B Req, B Ack, Parity Sync, B Output Enable**

These signals are used in the data transfer mode to control the handshake of data on Port B. Port B Req and Port B Ack are both gated with Port B Select.

5 The Port B Ack signal is used to strobe the data on the Port B data lines. The parity sync signal is used to indicate to a chip configured as the parity chip to indicate that the last words of data involved in the parity accumulation are on Port B. The Port B data  
10 lines will only be driven by the Fifo chip if all of the following conditions are met:

- a. the data transfer is from Port A to Port B;
- b. the Port B select signal is true;
- c. the Port B output enable signal is true; and
- 15 d. the chip is not configured as the parity chip or it is in parity correct mode and the Parity Sync signal is true.

**Reset**

This signal resets all the registers within the  
20 chip and causes all bidirectional pins to be in a high impedance state.

**DESCRIPTION OF OPERATION**

**Normal Operation.** Normally the chip acts as a simple FIFO chip. A FIFO is simulated by using two RAM  
25 buffers in a simple ping-pong mode. It is intended, but not mandatory, that data is burst into or out of

the FIFO on Port B. This is done by holding Port B Sel signal low and pulsing the Port B Ack signal. When transferring data from Port B to Port A, data is first written into RAM X and when this is full, the data paths will be switched such that Port B may start writing to RAM Y. Meanwhile the chip will begin emptying RAM X to Port A. When RAM Y is full and RAM X empty the data paths will be switched again such that Port B may reload RAM X and Port A may empty RAM Y.

5  
10     Port A Slave Mode. This is the default mode and the chip is reset to this condition. In this mode the chip waits for a master such as one of the SCSI adapter chips 542 to raise Port A Request for data transfer. If data is available the Fifo chip will respond with  
15     Port A Ack/Rdy.

20     Port A WD Mode. The chip may be configured to run in the WD or Western Digital mode. In this mode the chip must be configured as a slave on Port A. It differs from the default slave mode in that the chip responds with Read Enable or Write Enable as appropriate together with Port A Ack/Rdy. This mode is intended to allow the chip to be interfaced to the Western Digital 33C93A SCSI chip or the NCR 53C90 SCSI chip.

25     Port A Master Mode. When the chip is configured as a master, it will raise Port A Ack/Rdy when it is

ready for data transfer. This signal is expected to be tied to the Request input of a DMA controller which will respond with Port A Req when data is available. In order to allow the DMA controller to burst, the Port A Ack/Rdy signal will only be negated after every 8 or 16 bytes transferred.

Port B Parallel Write Mode. In parallel write mode, the chip is configured to be the parity chip for a parallel transfer from Port B to Port A. In this mode, when Port B Select and Port B Request are asserted, data is written into RAM X or RAM Y each time the Port B Ack signal is received. For the first block of 128 bytes data is simply copied into the selected RAM. The next 128 bytes driven on Port B will be exclusive-ORed with the first 128 bytes. This procedure will be repeated for all drives such that the parity is accumulated in this chip. The Parity Sync signal should be asserted to the parallel chip together with the last block of 128 bytes. This enables the chip to switch access to the other RAM and start accumulating a new 128 bytes of parity.

Port B Parallel Read Mode - Check Data. This mode is set if all drives are being read and parity is to be checked. In this case the Parity Correct bit in the Data Transfer Configuration Register is not set. The parity chip will first read 128 bytes on Port A as

in a normal read mode and then raise Port B Request. While it has this signal asserted the chip will monitor the Port B Ack signals and exclusive-or the data on Port B with the data in its selected RAM. The Parity Sync should again be asserted with the last block of 128 bytes. In this mode the chip will not drive the Port B data lines but will check the output of its exclusive-or logic for zero. If any bits are set at this time a parallel parity error will be flagged.

5  
10        Port B Parallel Read Mode - Correct Data. This mode is set by setting the Parity Correct bit in the Data Transfer Configuration Register. In this case the chip will work exactly as in the check mode except that when Port B Output Enable, Port B Select and Parity Sync are true the data is driven onto the Port B data lines and a parallel parity check for zero is not performed.

15  
20        Byte Swap. In the normal mode it is expected that Port B bits 00-07 are the first byte, bits 10-17 the second byte, bits 20-27 the third byte, and bits 30-37 the last byte of each word. The order of these bytes may be changed by writing to the byte swap bits in the configuration register such that the byte address bits are inverted. The way the bytes are written and read also depend on whether the CPU  
25 interface is configured as 16 or 8 bits. The following

table shows the byte alignments for the different possibilities for data transfer using the Port A Request / Acknowledge handshake:

	CPU I/F	Invert Addr 1	Invert Addr 0	Port B 00-07	Port B 10-17	Port B 20-27	Port B 30-37
5	8	False	False	Port A byte 0	Port A byte 1	Port A byte 2	Port A byte 1
10	8	False	True	Port A byte 1	Port A byte 0	Port A byte 3	Port A byte 2
	8	True	False	Port A byte 2	Port A byte 3	Port A byte 0	Port A byte 1
	8	True	True	Port A byte 3	Port A byte 2	Port A byte 1	Port A byte 0
15	16	False	False	Port A byte 0	uProc byte 0	Port A byte 1	uProc byte 1
	16	False	True	uProc byte 0	Port A byte 0	uProc byte 1	Port A byte 1
	16	True	False	Port A byte 1	uProc byte 1	Port A byte 0	uProc byte 0
20	16	True	True	uProc byte 1	Port A byte 1	uProc byte 0	Port A byte 0

When the Fifo is accessed by reading or writing the Fifo Data Register through the microprocessor port in 8 bit mode, the bytes are in the same order as the table above but the uProc data port is used instead of Port A. In 16 bit mode the table above applies.

Odd Length Transfers. If the data transfer is not a multiple of 32 words, or 128 bytes, the microprocessor must manipulate the internal registers of the chip to ensure all data is transferred. Port A Ack and Port B Req are normally not asserted until all

32 words of the selected RAM are available. These signals may be forced by writing to the appropriate RAM status bits of the Data Transfer Status Register.

5 When an odd length transfer has taken place the microprocessor must wait until both ports are quiescent before manipulating any registers. It should then reset both of the Enable Data Transfer bits for Port A and Port B in the Data Transfer Control Register. It must then determine by reading their Address Registers  
10 and the RAM Access Control Register whether RAM X or RAM Y holds the odd length data. It should then set the corresponding Address Register to a value of 20 hexadecimal, forcing the RAM full bit and setting the address to the first word. Finally the microprocessor  
15 should set the Enable Data Transfer bits to allow the chip to complete the transfer.

At this point the Fifo chip will think that there are now a full 128 bytes of data in the RAM and will transfer 128 bytes if allowed to do so. The fact that  
20 some of these 128 bytes are not valid must be recognized externally to the FIFO chip.

PROGRAMMABLE REGISTERS

Data Transfer Configuration Register (Read/Write)

Register Address 0. This register is cleared by the reset signal.

- 5           Bit 0    WD Mode. Set if data transfers are to use the Western Digital WD33C93A protocol, otherwise the Adaptec 6250 protocol will be used.
- 10           Bit 1    Parity Chip. Set if this chip is to accumulate Port B parities.
- Bit 2    Parity Correct Mode. Set if the parity chip is to correct parallel parity on Port B.
- 15           Bit 3    CPU Interface 16 bits wide. If set, the microprocessor data bits are combined with the Port A data bits to effectively produce a 16 bit Port. All accesses by the microprocessor as well as all data transferred using the Port A Request and Acknowledge handshake will transfer 16 bits.
- 20           Bit 4    Invert Port A byte address 0. Set to invert the least significant bit of Port A byte address.
- 25           Bit 5    Invert Port A byte address 1. Set to invert the most significant bit of Port A byte address.
- Bit 6    Checksum Carry Wrap. Set to enable the carry out of the 16 bit checksum adder to carry back into the least significant bit of the adder.
- 30           Bit 7    Reset. Writing a 1 to this bit will reset the other registers. This bit resets itself after a maximum of 2 clock cycles and will therefore normally be read as a 0. No other register should be written for a minimum of 4 clock cycles after writing to this bit.
- 35

Data Transfer Control Register (Read/Write)

Register Address 1. This register is cleared by the reset signal or by writing to the reset bit.

- 5 Bit 0 Enable Data Transfer on Port A. Set to enable the Port A Req/Ack handshake.
- 10 Bit 1 Enable Data Transfer on Port B. Set to enable the Port B Req/Ack handshake.
- 15 Bit 2 Port A to Port B. If set, data transfer is from Port A to Port B. If reset, data transfer is from Port B to Port A. In order to avoid any glitches on the request lines, the state of this bit should not be altered at the same time as the enable data transfer bits 0 or 1 above.
- 20 Bit 3 uProcessor Parity Enable. Set if parity is to be checked on the microprocessor interface. It will only be checked when writing to the Fifo Data Register or reading from the Fifo Data or Checksum Registers, or during a Port A Request/Acknowledge transfer in 16 bit mode. The chip will, however, always re-generate parity ensuring that correct parity is written to the RAM or read on the microprocessor interface.
- 25 Bit 4 Port A Parity Enable. Set if parity is to be checked on Port A. It is checked when accessing the Fifo Data Register in 16 bit mode, or during a Port A Request/Acknowledge transfer. The chip will, however, always re-generate parity ensuring that correct parity is written to the RAM or read on the Port A interface.
- 30 Bit 5 Port B Parity Enable. Set if Port B data has valid byte parities. If it is not set, byte parity is generated internally to the chip when writing to the RAMs. Byte parity is not checked when writing from Port B, but always checked when reading to Port B.
- 35
- 40



- 5 Bit 6 Checksum Enable. Set to enable writing to the 16 bit checksum register. This register accumulates a 16 bit checksum for all RAM accesses, including accesses to the Fifo Data Register, as well as all writes to the checksum register. This bit must be reset before reading from the Checksum Register.
- 10 Bit 7 Port A Master. Set if Port A is to operate in the master mode on Port A during the data transfer.

Data Transfer Status Register (Read Only)

Register Address 2. This register is cleared by the reset signal or by writing to the reset bit.

- 15 Bit 0 Data in RAM X or RAM Y. Set if any bits are true in the RAM X, RAM Y, or Port A byte address registers.
- 20 Bit 1 uProc Port Parity Error. Set if the uProc Parity Enable bit is set and a parity error is detected on the microprocessor interface during any RAM access or write to the Checksum Register in 16 bit mode.
- 25 Bit 2 Port A Parity Error. Set if the Port A Parity Enable bit is set and a parity error is detected on the Port A interface during any RAM access or write to the Checksum Register.
- 30 Bit 3 Port B Parallel Parity Error. Set if the chip is configured as the parity chip, is not in parity correct mode, and a non zero result is detected when the Parity Sync signal is true. It is also set whenever data is read out onto Port B and the data being read back through the bidirectional buffer does not compare.
- 35
- 40 Bits 4-7 Port B Bytes 0-3 Parity Error. Set whenever the data being read out of the RAMs on the Port B side has bad parity.

Ram Access Control Register (Read/Write)

Register Address 3. This register is cleared by the reset signal or by writing to the reset bit. The Enable Data Transfer bits in the Data Transfer Control Register must be reset before attempting to write to this register, else the write will be ignored.

- 5
- 10 Bit 0 Port A byte address 0. This bit is the least significant byte address bit. It is read directly bypassing any inversion done by the invert bit in the Data Transfer Configuration Register.
- 15 Bit 1 Port A byte address 1. This bit is the most significant byte address bit. It is read directly bypassing any inversion done by the invert bit in the Data Transfer Configuration Register.
- 20 Bit 2 Port A to RAM Y. Set if Port A is accessing RAM Y, and reset if it is accessing RAM X .
- 25 Bit 3 Port B to RAM Y. Set if Port B is accessing RAM Y, and reset if it is accessing RAM X .
- 30 Bit 4 Long Burst. If the chip is configured to transfer data on Port A as a master, and this bit is reset, the chip will only negate Port A Ack/Rdy after every 8 bytes, or 4 words in 16 bit mode, have been transferred. If this bit is set, Port A Ack/Rdy will be negated every 16 bytes, or 8 words in 16 bit mode.
- Bits 5-7 Not Used.

RAM X Address Register (Read/Write)

Register Address 4. This register is cleared by the reset signal or by writing to the reset bit. The Enable Data Transfer bits in the Data Transfer Control

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Register must be reset before attempting to write to this register, else the write will be ignored.

Bits 0-4           RAM X word address  
Bit    5           RAM X full  
5       Bits 6-7       Not Used

RAM Y Address Register (Read/Write)

Register Address 5. This register is cleared by the reset signal or by writing to the reset bit. The Enable Data Transfer bits in the Data Transfer Control Register must be reset before attempting to write to this register, else the write will be ignored.

10       Bits 0-4           RAM Y word address  
          Bit    5           RAM Y full  
          Bits 6-7       Not Used

15       Fifo Data Register (Read/Write)

Register Address 6. The Enable Data Transfer bits in the Data Transfer Control Register must be reset before attempting to write to this register, else the write will be ignored. The Port A to Port B bit in the Data Transfer Control register must also be set before writing this register. If it is not, the RAM controls will be incremented but no data will be written to the RAM. For consistency, the Port A to Port B should be reset prior to reading this register.

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

Bits 0-7 are Fifo Data. The microprocessor may access the FIFO by reading or writing this register. The RAM control registers are updated as if the access was using Port A. If the chip is configured with a 16 bit CPU Interface the most significant byte will use the Port A 0-7 data lines, and each Port-A access will increment the Port A byte address by 2.

Port A Checksum Register (Read/Write)

Register Address 7. This register is cleared by the reset signal or by writing to the reset bit.

Bits 0-7 are Checksum Data. The chip will accumulate a 16 bit checksum for all Port A accesses. If the chip is configured with a 16 bit CPU interface, the most significant byte is read on the Port A 0-7 data lines. If data is written directly to this register it is added to the current contents rather than overwriting them. It is important to note that the Checksum Enable bit in the Data Transfer Control Register must be set to write this register and reset to read it.

PROGRAMMING THE FIFO CHIP

In general the fifo chip is programmed by writing to the data transfer configuration and control registers to enable a data transfer, and by reading

the data transfer status register at the end of the transfer to check the completion status. Usually the data transfer itself will take place with both the Port A and the Port B handshakes enabled, and in this case  
5 the data transfer itself should be done without any other microprocessor interaction. In some applications, however, the Port A handshake may not be enabled, and it will be necessary for the microprocessor to fill or empty the fifo by repeatedly  
10 writing or reading the Fifo Data Register.

Since the fifo chip has no knowledge of any byte counts, there is no way of telling when any data transfer is complete by reading any register within this chip itself. Determination of whether the data  
15 transfer has been completed must therefore be done by some other circuitry outside this chip.

The following C language routines illustrate how the parity FIFO chip may be programmed. The routines assume that both Port A and the microprocessor port are  
20 connected to the system microprocessor, and return a size code of 16 bits, but that the hardware addresses the Fifo chip as long 32 bit registers.

```
struct FIFO_regs (
  unsigned char config,a1,a2,a3 ;
  unsigned char control,b1,b2,b3;
  unsigned char status,c1,c2,c3;
5   unsigned char ram_access_control,d1,d2,d3;
  unsigned char ram_X_addr,e1,e2,e3;
  unsigned char ram_Y_addr,f1,f2,f3;
  unsigned long data;
  unsigned int checksum,h1;
10  );

#define FIFO1 ((struct FIFO_regs*) FIFO_BASE_ADDRESS)

#define FIFO_RESET 0x80
#define FIFO_16_BITS 0x08
#define FIFO_CARRY_WRAP 0x40
15 #define FIFO_PORT_A_ENABLE 0x01
#define FIFO_PORT_B_ENABLE 0x02
#define FIFO_PORT_ENABLES 0x03
#define FIFO_PORT_A_TO_B 0x04
#define FIFO_CHECKSUM_ENABLE 0x40
20 #define FIFO_DATA_IN_RAM 0x01
#define FIFO_FORCE_RAM_FULL 0x20

#define PORT_A_TO_PORT_B(fifo) ((fifo-> control ) & 0x04)
#define PORT_A_BYTE_ADDRESS(fifo) ((fifo->ram_access_control) &
  0x03)
25 #define PORT_A_TO_RAM_Y(fifo) ((fifo->ram_access_control) &
  0x04)
#define PORT_B_TO_RAM_Y(fifo) ((fifo-> ram_access_control ) &
  0x08)

/*****
30   The following routine initiates a Fifo data transfer using
  two values passed to it.

  config_data   This is the data to be written to the
                 configuration register.

  control_data  This is the data to be written to the Data
35   Transfer Control Register.  If the data transfer
                 is to take place automatically using both the
                 Port A and Port B handshakes, both data transfer
                 enables bits should be set in this parameter.
  *****/

40 FIFO_initiate_data_transfer(config_data, control_data)
   unsigned char config_data, control_data;
   {
     FIFO1->config = config_data | FIFO_RESET;    /* Set
           Configuration value & Reset */

```

```
FIFO1->control = control_data & (~FIFO_PORT_ENABLES); /* Set
everything but enables */
FIFO1->control = control_data ; /* Set data transfer
enables */
5 }

/*****
The following routine forces the transfer of any odd bytes
that have been left in the Fifo at the end of a data transfer.
It first disables both ports, then forces the Ram Full bits, and
then re-enables the appropriate Port.
10 *****/

FIFO_force_odd_length_transfer()
{
FIFO1->control &= ~FIFO_PORT_ENABLES; /* Disable Ports A & B */
15 if (PORT_A_TO_PORT_B(FIFO1)) {
if (PORT_A_TO_RAM_Y(FIFO1)) {
FIFO1->ram_Y_addr = FIFO_FORCE_RAM_FULL; /* Set RAM Y
full */
}
else FIFO1->ram_X_addr = FIFO_FORCE_RAM_FULL ; /* Set RAM
20 X full */
FIFO1->control |= FIFO_PORT_B_ENABLE ; /* Re-Enable
Port B */
}
25 else {
if (PORT_B_TO_RAM_Y(FIFO1)) {
FIFO1->ram_Y_addr = FIFO_FORCE_RAM_FULL ; /* Set
RAM Y full */
}
else FIFO1->ram_X_addr = FIFO_FORCE_RAM_FULL ; /* Set RAM
30 X full */
FIFO1->control |= FIFO_PORT_A_ENABLE ; /* Re-Enable
Port A */
}
35 }

/*****
The following routine returns how many odd bytes have been
left in the Fifo at the end of a data transfer.
*****/

40 int FIFO_count_odd_bytes()
{
int number_odd_bytes;
number_odd_bytes=0;
if (FIFO1->status & FIFO_DATA_IN_RAM) {
45 if (PORT_A_TO_PORT_B(FIFO1)) {
number_odd_bytes = (PORT_A_BYTE_ADDRESS(FIFO1)) ;
if (PORT_A_TO_RAM_Y(FIFO1))
number_odd_bytes += (FIFO1->ram_Y_addr) * 4 ;
}
```

```

    else number_odd_bytes += (FIFO1->ram_X_addr) * 4 ;
}
else {
5   if (PORT_B_TO_RAM_Y(FIFO1))
        number_odd_bytes = (FIFO1->ram_Y_addr) * 4 ;
        else number_odd_bytes = (FIFO1->ram_X_addr) * 4 ;
}
}
return (number_odd_bytes);
10 }

/*****
The following routine tests the microprocessor interface of
the chip. It first writes and reads the first 6 registers. It
then writes 1s, 0s, and an address pattern to the RAM, reading the
15 data back and checking it.

The test returns a bit significant error code where each
bit represents the address of the registers that failed.

Bit 0 = config register failed
Bit 1 = control register failed
20 Bit 2 = status register failed
Bit 3 = ram access control register failed
Bit 4 = ram X address register failed
Bit 5 = ram Y address register failed
Bit 6 = data register failed
25 Bit 7 = checksum register failed
*****/

#define RAM_DEPTH 64      /* number of long words in Fifo Ram */

reg_expected_data[6] = { 0x7F, 0xFF, 0x00, 0x1F, 0x3F, 0x3F };

char FIFO_uprocessor_interface_test()
30 {
    unsigned long test_data;
    char *register_addr;
    int i;
    char j,error;
35   FIFO1->config = FIFO_RESET;      /* reset the chip */
    error=0;
    register_addr =(char *) FIFO1;
    j=1;

40   /* first test registers 0 thru 5 */

    for (i=0; i<6; i++) {
        *register_addr = 0xFF;      /* write test data */
        if (*register_addr != reg_expected_data[i]) error |= j;
45   *register_addr = 0;      /* write 0s to register */
        if (*register_addr) error |= j;
    }
}

```



```

5      *register_addr = 0xFF;          /* write test data again */
      if (*register_addr != reg_expected_data[i]) error |= j;
      FIFO1->config = FIFO_RESET;    /* reset the chip */
      if (*register_addr) error |= j; /* register should be 0 */
      register_addr++;              /* go to next register */
      j <<= 1;
    }

    /* now test Ram data & checksum registers
10    test 1s throughout Ram & then test 0s */

    for (test_data = -1; test_data != 1; test_data++) { /* test
for 1s & 0s */
      FIFO1->config = FIFO_RESET | FIFO_16_BITS ;
15      FIFO1->control = FIFO_PORT_A_TO_B;
      for (i=0; i<RAM_DEPTH; i++) /* write data to RAM */
          FIFO1->data = test_data;
      FIFO1->control = 0;
      for (i=0; i<RAM_DEPTH; i++)
20          if (FIFO1->data != test_data) error |= j; /* read
& check data */
      if (FIFO1->checksum) error |= 0x80; /* checksum
should = 0 */
    }
25

    /* now test Ram data with address pattern
    uses a different pattern for every byte */

    test_data=0x00010203; /* address pattern start */
    FIFO1->config = FIFO_RESET | FIFO_16_BITS | FIFO_CARRY_WRAP;
30    FIFO1->control = FIFO_PORT_A_TO_B | FIFO_CHECKSUM_ENABLE;
    for (i=0; i<RAM_DEPTH; i++) {
        FIFO1->data = test_data; /* write address pattern */
        test_data += 0x04040404;
    }
35    test_data=0x00010203; /* address pattern start */
    FIFO1->control = FIFO_CHECKSUM_ENABLE;
    for (i=0; i<RAM_DEPTH; i++) {
        if (FIFO1->status != FIFO_DATA_IN_RAM)
            error |= 0x04; /* should be data in ram */
40        if (FIFO1->data != test_data) error |= j; /* read &
check address pattern */
        test_data += 0x04040404;
    }
    if (FIFO1->checksum != 0x0102) error |= 0x80; /* test
45    checksum of address pattern */
    FIFO1->config = FIFO_RESET | FIFO_16_BITS ; /* inhibit carry
wrap */
    FIFO1->checksum = 0xFEFE; /* writing adds to checksum */

```

-132-

```
if (FIFO1->checksum) error !=0x80; /* checksum should be 0 */  
if (FIFO1->status) error != 0x04; /* status should be 0 */  
return (error);  
}
```

Attorney Docket No.:AUSP7209  
WP1/WSW/AUSP/7209.001

8/24/89-7

CLAIMS

1. Network server apparatus for use with a data network and a mass storage device, comprising:

an interface processor unit coupleable to said network and to said mass storage device;

a host processor unit;

means in said interface processor unit for satisfying requests from said network to store data from said network on said mass storage device;

means in said interface processor unit for satisfying requests from said network to retrieve data from said mass storage device to said network;

means in said interface processor unit for satisfying requests from said host processor unit to store data from said host processor unit on said mass storage device; and

means in said interface processor unit for satisfying requests from said host processor unit to retrieve data from said mass storage device to said host processor unit.

2. Apparatus according to claim 1, wherein said interface processor unit comprises:

a network control unit coupleable to said network;

a data control unit coupleable to said mass storage device;

a buffer memory;

means in said network control unit for transmitting to said data control unit requests from said network to store specified storage data from said network on said mass storage device;

means in said network control unit for transmitting said specified storage data from said network to said buffer memory and from said buffer memory to said data control unit;

means in said network control unit for transmitting to said data control unit requests from said network to retrieve specified retrieval data from said mass storage device to said network; and

means in said network control unit for transmitting said specified retrieval data from said data control unit to said buffer memory and from said buffer memory to said network.

3. Apparatus according to claim 2, wherein said data control unit comprises:

a storage processor unit coupleable to said mass storage device;

a file processor unit;

means on said file processor unit for translating said file system level storage requests from said network into requests to store data at specified physical storage locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to write data from said buffer memory into said specified physical storage locations in said mass storage device;

means on said file processor unit for translating file system level retrieval requests from said network into requests to retrieve data from specified physical retrieval locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to retrieve data from said specified physical retrieval locations in said mass storage device to said buffer memory if said data from said specified physical locations is not already in said buffer memory; and

means in said storage processor unit for transmitting data between said buffer memory and said mass storage device.

4. Apparatus according to claim 1, for use further with a buffer memory, and wherein said requests from said network to store and retrieve data include file system level storage and retrieval requests respectively, and wherein said interface processor unit comprises:

a storage processor unit coupleable to said mass storage device;

a file processor unit;

- 135 -

Attorney Docket No.: AUSP7209(IL)MCF/GBR/MSW  
msw/ausp/7209il.001

means on said file processor unit for translating said file system level storage requests into requests to store data at specified physical storage locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to write data from said buffer memory into said specified physical storage locations in said mass storage device;

means on said file processor unit for translating said file system level retrieval requests into requests to retrieve data from specified physical retrieval locations in said mass storage device;

means on said file processor unit for instructing said storage processor unit to retrieve data from said specified physical retrieval locations in said mass storage device to said buffer memory if said data from said specified physical locations is not already in said buffer memory; and

means in said storage processor unit for transmitting data between said buffer memory and said mass storage device.

5. Network server apparatus for use with a data network, comprising:

a network controller coupleable to said network to receive incoming information packets over said network, said incoming information packets including certain packets which contain part or all of a request to said

server apparatus, said request being in either a first or a second class of requests to said server apparatus;

a first additional processor;

an interchange bus different from said network and coupled between said network controller and said first additional processor;

means in said network controller for detecting and satisfying requests in said first class of requests contained in said certain incoming information packets, said network controller lacking means in said network controller for satisfying requests in said second class of requests; and

means in said network controller for satisfying requests received over said interchange bus from said first additional processor.

6. Apparatus according to claim 5, wherein said means in said network controller for detecting and satisfying requests in said first class of requests, assembles said requests in said first class of requests into assembled requests before satisfying said requests in said first class of requests.

7. Apparatus according to claim 5, wherein said packets each include a network node destination address, wherein said means in said network controller for detecting and satisfying requests in said first class of requests, assembles said requests in said first class of

requests, in a format which omits said network node destination addresses, before satisfying said requests in said first class of requests.

8. Apparatus according to claim 5, wherein said means in said network controller for detecting and satisfying requests in said first class includes means for preparing an outgoing message in response to one of said first class of requests, means for packaging said outgoing message in outgoing information packets suitable for transmission over said network, and means for transmitting said outgoing information packets over said network.

9. Apparatus according to claim 5, wherein said first class of requests comprises requests for an address of said server apparatus, and wherein said means in said network controller for detecting and satisfying requests in said first class comprises means for preparing a response packet to such an address request and means for transmitting said response packet over said network.

10. Apparatus according to claim 5, for use further with a second data network, said network controller being coupleable further to said second network, wherein said first class of requests comprises requests to route a message to a destination reachable over said second network, and wherein said means in said



network controller for detecting and satisfying requests in said first class comprises means for detecting that one of said certain packets comprises a request to route a message contained in said one of said certain packets to a destination reachable over said second network, and means for transmitting said message over said second network.

11. Apparatus according to claim 10, for use further with a third data network, said network controller further comprising means in said network controller for detecting particular requests in said incoming information packets to route a message contained in said particular requests, to a destination reachable over said third network, said apparatus further comprising:

a second network controller coupled to said interchange bus and coupleable to said third data network;

means for delivering said message contained in said particular requests to said second network controller over said interchange bus; and

means in said second network controller for transmitting said message contained in said particular requests over said third network.

12. Apparatus according to claim 5, for use further with a third data network, said network

controller further comprising means in said network controller for detecting particular requests in said incoming information packets to route a message contained in said particular requests, to a destination reachable over said third network, said apparatus further comprising:

a second network controller coupled to said interchange bus and coupleable to said third data network;

means for delivering said message contained in said particular requests to said second network controller over said interchange bus; and

means in said second network controller for transmitting said message contained in said particular requests over said third network.

13. Apparatus according to claim 5, for use further with a mass storage device, wherein said first additional processor comprises a data control unit coupleable to said mass storage device, wherein said second class of requests comprises remote calls to procedures for managing a file system in said mass storage device, and wherein said means in said first additional processor for further processing said assembled requests in said second class of requests comprises means for executing file system procedures on

said mass storage device in response to said assembled requests.

14. Apparatus according to claim 13, wherein said file system procedures include a read procedure for reading data from said mass storage device,

said means in said first additional processor for further processing said assembled requests including means for reading data from a specified location in said mass storage device in response to a remote call to said read procedure,

said apparatus further including means for delivering said data to said network controller,

said network controller further comprising means on said network controller for packaging said data in outgoing information packets suitable for transmission over said network, and means for transmitting said outgoing information packets over said network.

15. Apparatus according to claim 14, wherein said means for delivering comprises:

a system buffer memory coupled to said interchange bus;

means in said data control unit for transferring said data over said interchange bus into said buffer memory; and

means in said network controller for transferring said data over said interchange bus from said system buffer memory to said network controller.

16. Apparatus according to claim 13, wherein said file system procedures include a read procedure for reading a specified number of bytes of data from said mass storage device beginning at an address specified in logical terms including a file system ID and a file ID, said means for executing file system procedures comprising:

means for converting the logical address specified in a remote call to said read procedure to a physical address; and

means for reading data from said physical address in said mass storage device.

17. Apparatus according to claim 16, wherein said mass storage device comprises a disk drive having a numbered tracks and sectors, wherein said logical address specifies said file system ID, said file ID, and a byte offset, and wherein said physical address specifies a corresponding track and sector number.

18. Apparatus according to claim 13, wherein said file system procedures include a read procedure for reading a specified number of bytes of data from said mass storage device beginning at an address specified in logical terms including a file system ID and a file ID,

said data control unit comprising a file processor coupled to said interchange bus and a storage processor coupled to said interchange bus and coupleable to said mass storage device,

said file processor comprising means for converting the logical address specified in a remote call to said read procedure to a physical address,

said apparatus further comprising means for delivering said physical address to said storage processor,

said storage processor comprising means for reading data from said physical address in said mass storage device and for transferring said data over said interchange bus into said buffer memory; and

means in said network controller for transferring said data over said interchange bus from said system buffer memory to said network controller.

19. Apparatus according to claim 13, wherein said file system procedures include a write procedure for writing data contained in an assembled request, to said mass storage device,

said means in said first additional processor for further processing said assembled requests including means for writing said data to a specified location in said mass storage device in response to a remote call to said read procedure.

- 143 -

Attorney Docket No.: AUSP7209(IL)MCF/GBR/MSW  
msw/ausp/720911.001

20. Apparatus according to claim 5, wherein said network controller comprises:

a microprocessor;

a local instruction memory containing local instruction code;

a local bus coupled between said microprocessor and said local instruction memory;

bus interface means for interfacing said microprocessor with said interchange bus at times determined by said microprocessor in response to said local instruction code; and

network interface means for interfacing said microprocessor with said data network,

said local instruction memory including all instruction code necessary for said microprocessor to perform said function of detecting and satisfying requests in said first class of requests.

21. Network server apparatus for use with a data network, comprising:

a network controller coupleable to said network to receive incoming information packets over said network, said incoming information packets including certain packets which contain part or all of a message to said server apparatus, said message being in either a first or a second class of messages to said server apparatus,

said messages in said first class of messages including certain messages containing requests;

a host computer;

an interchange bus different from said network and coupled between said network controller and said host computer;

means in said network controller for detecting and satisfying said requests in said first class of messages; and

means for satisfying requests received over said interchange bus from said host computer.

22. Apparatus according to claim 21, wherein said means in said network controller for detecting and satisfying requests in said first class includes means for preparing an outgoing message in response to one of said requests in said first class of messages, means for packaging said outgoing message in outgoing information packets suitable for transmission over said network, and means for transmitting said outgoing information packets over said network.

23. Apparatus according to claim 21, for use further with a second data network, said network controller being coupleable further to said second network, wherein said first class of messages comprises messages to be routed to a destination reachable over said second network, and wherein said means in said

- 145 -

Attorney Docket No.: AUSP7209(IL)MCF/GBR/USV  
USM/ausp/7209il.001

network controller for detecting and satisfying requests in said first class comprises means for detecting that one of said certain packets includes a request to route a message contained in said one of said certain packets to a destination reachable over said second network, and means for transmitting said message over said second network.

24. Apparatus according to claim 21, for use further with a third data network, said network controller further comprising means in said network controller for detecting particular messages in said incoming information packets to be routed to a destination reachable over said third network, said apparatus further comprising:

a second network controller coupled to said interchange bus and coupleable to said third data network;

means for delivering said particular messages to said second network controller over said interchange bus, substantially without involving said host computer; and

means in said second network controller for transmitting said message contained in said particular requests over said third network, substantially without involving said host computer.

- 146 -

Attorney Docket No.: AUSP7209(1L)MCF/GBR/MSW  
msw/ausp/7209it.001



25. Apparatus according to claim 21, for use further with a mass storage device, further comprising a data control unit coupleable to said mass storage device,

said network controller further comprising means in said network controller for detecting ones of said incoming information packets containing remote calls to procedures for managing a file system in said mass storage device, and means in said network controller for assembling said remote calls from said incoming packets into assembled calls, substantially without involving said host computer,

said apparatus further comprising means for delivering said assembled file system calls to said data control unit over said interchange bus substantially without involving said host computer, and said data control unit comprising means in said data control unit for executing file system procedures on said mass storage device in response to said assembled file system calls, substantially without involving said host computer.

26. Apparatus according to claim 21, wherein said network controller comprises:

a microprocessor;

a local instruction memory containing local instruction code;

- 147 -

Attorney Docket No.: AUSP7209(IL)MCF/GBR/VSU  
MSW/ausp/7209i1.001

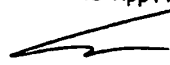
a local bus coupled between said microprocessor and said local instruction memory;

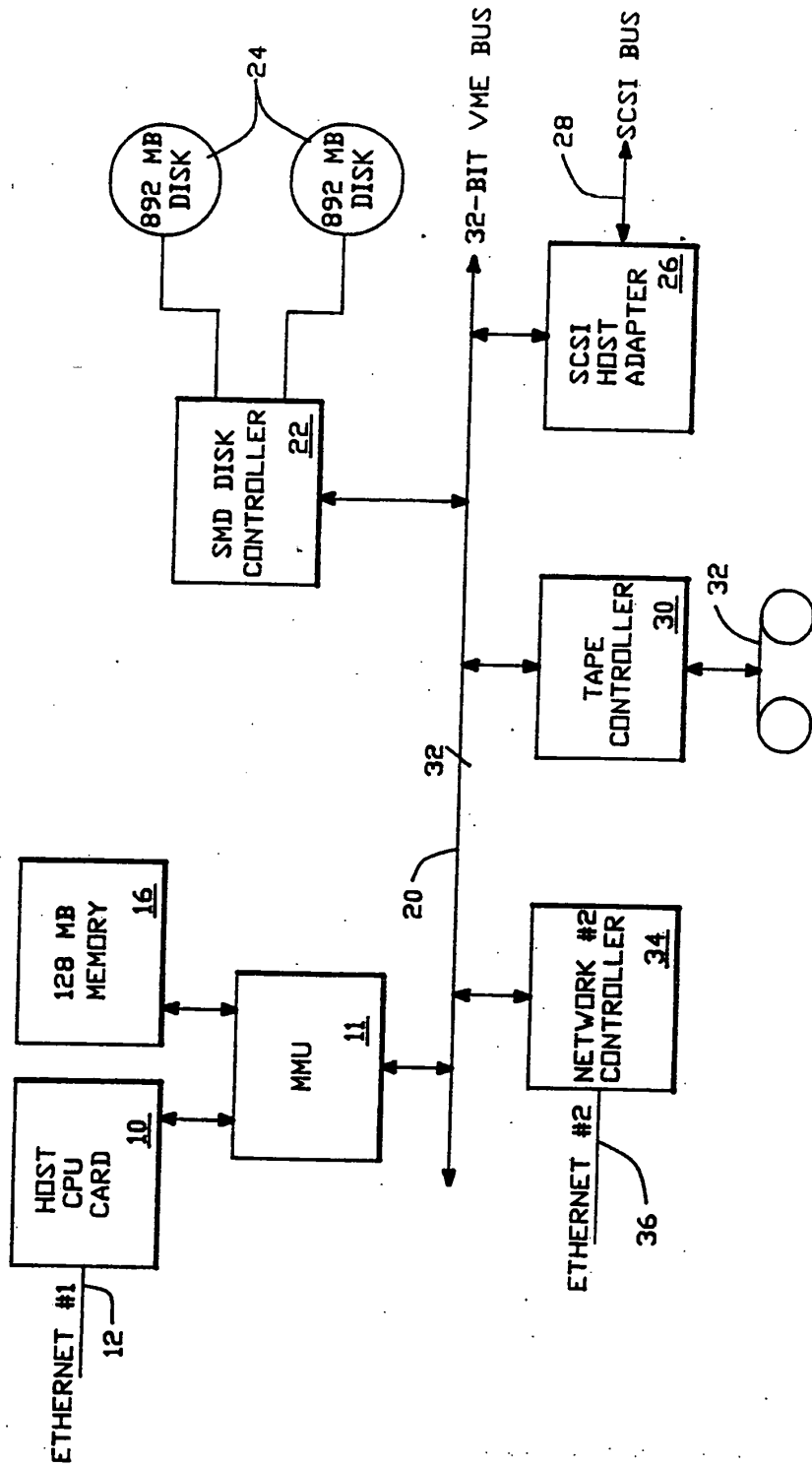
bus interface means for interfacing said microprocessor with said interchange bus at times determined by said microprocessor in response to said local instruction code; and

network interface means for interfacing said microprocessor with said data network,

said local instruction memory including all instruction code necessary for said microprocessor to perform said function of detecting and satisfying requests in said first class of requests.

For the Applicant,

  
Sanford T. Colb & Co.  
C:17980



(PRIOR ART)

FIG.-1

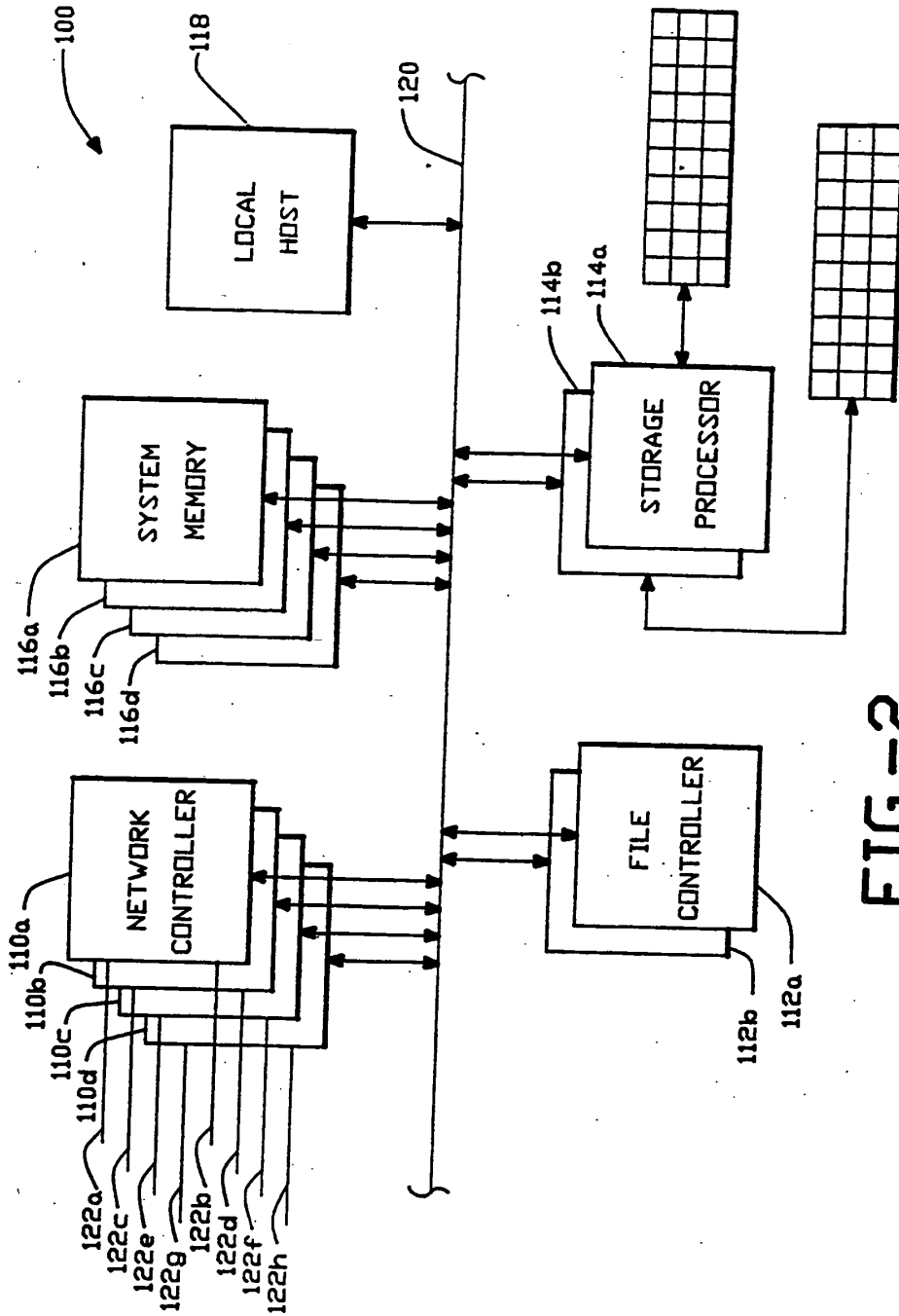


FIG.-2

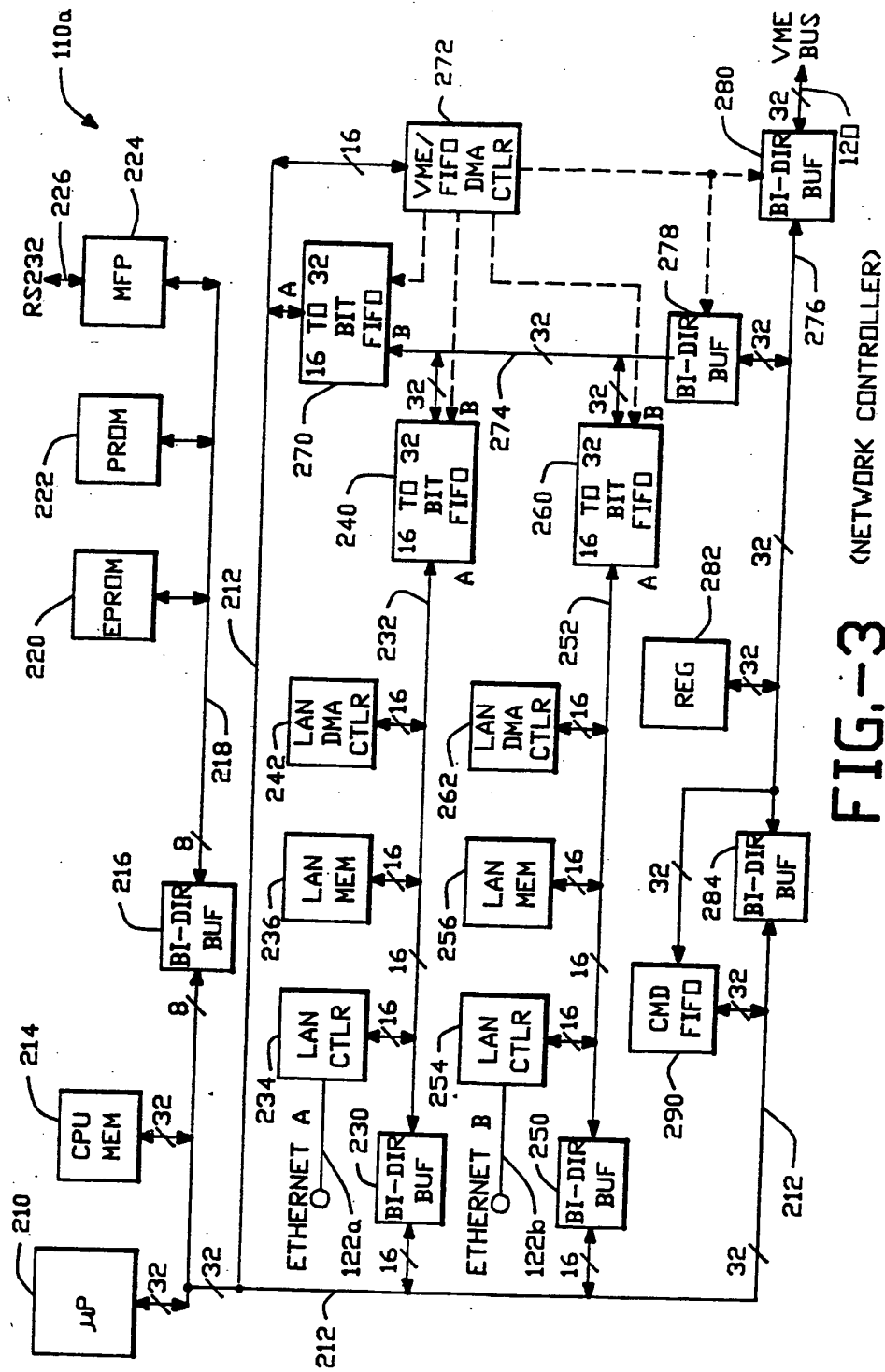
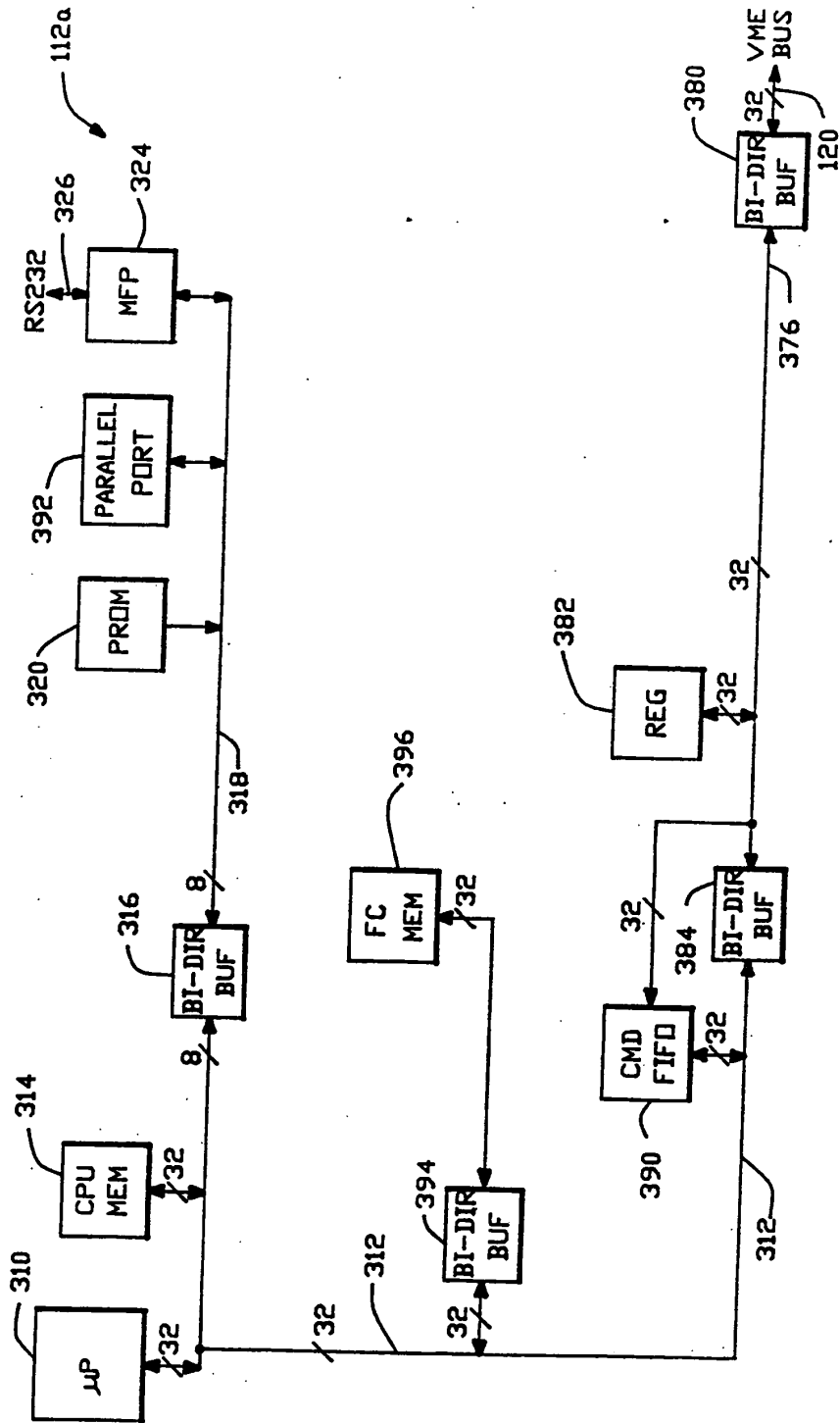


FIG. 3  
(NETWORK CONTROLLER)

AUSPEX SYSTEMS, INC.

TWELVE SHEETS  
SHEET NO. FOUR



(FILE CONTROLLER)

FIG.-4

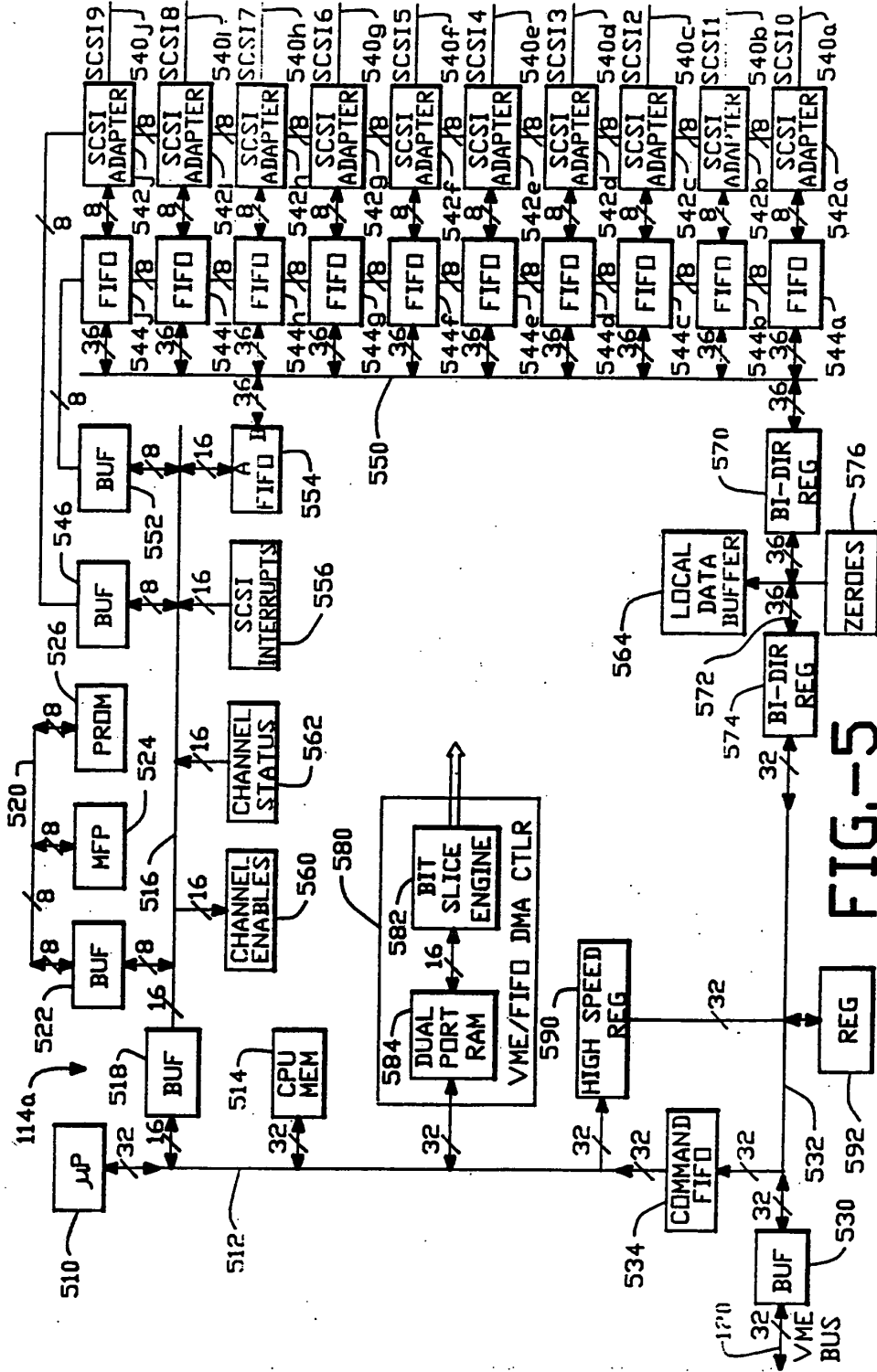
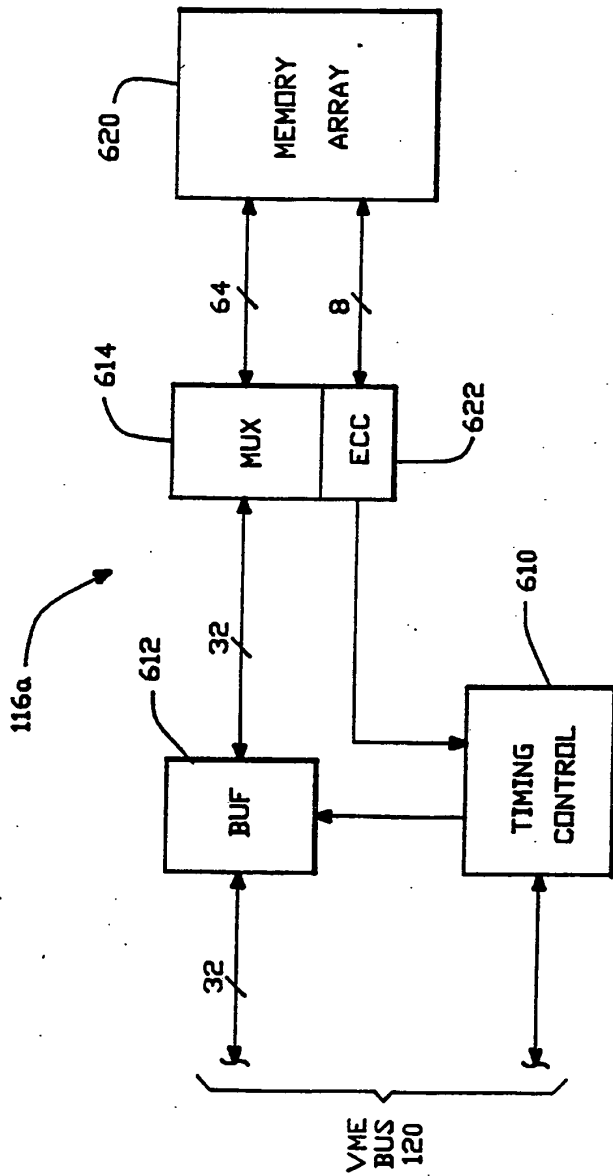


FIG.-5



(SYSTEM MEMORY)

FIG.-6



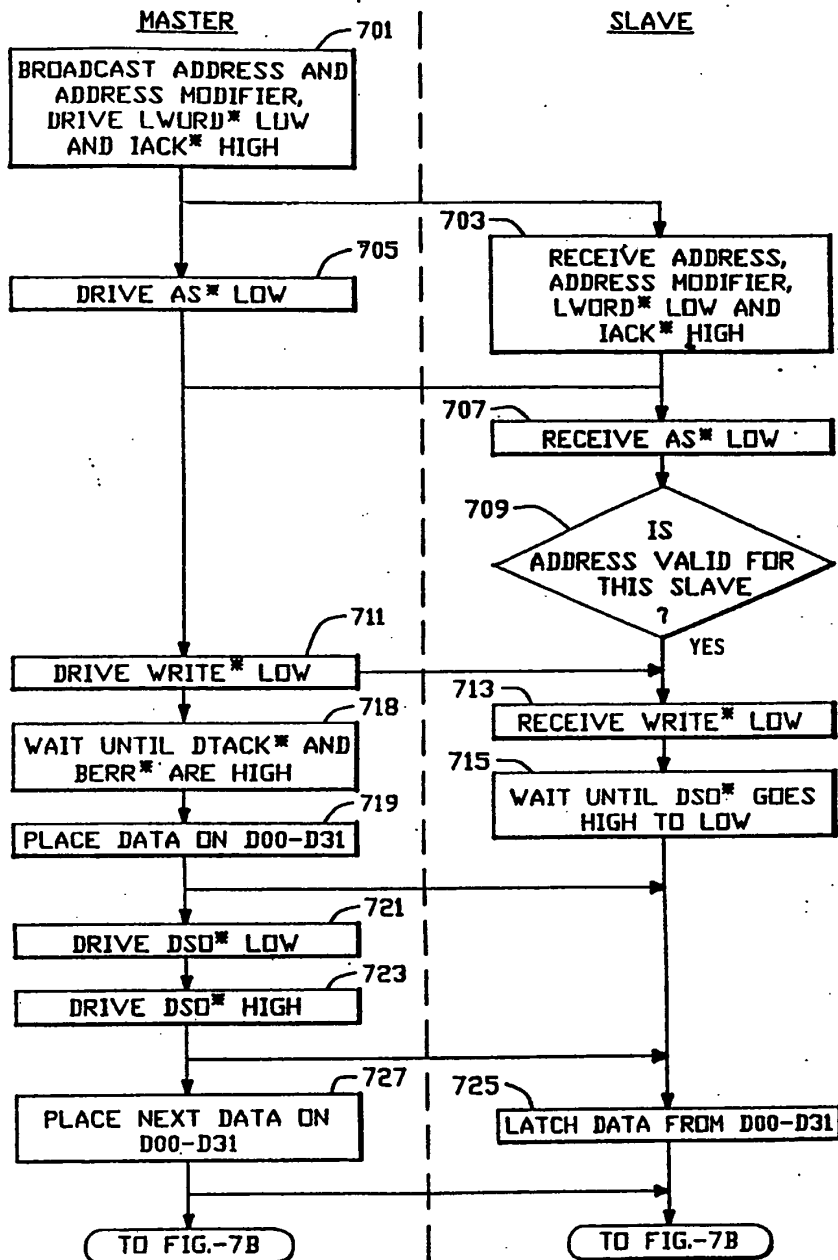


FIG.-7A

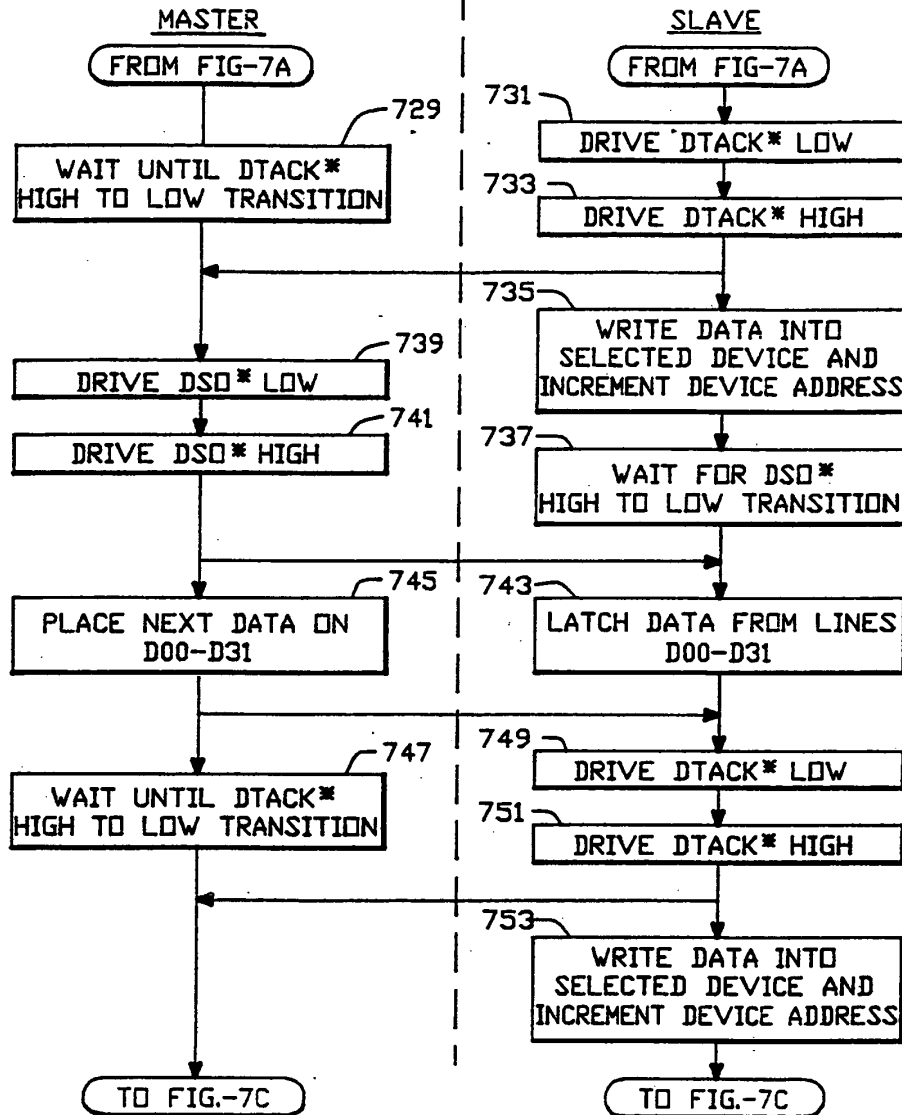


FIG.-7B

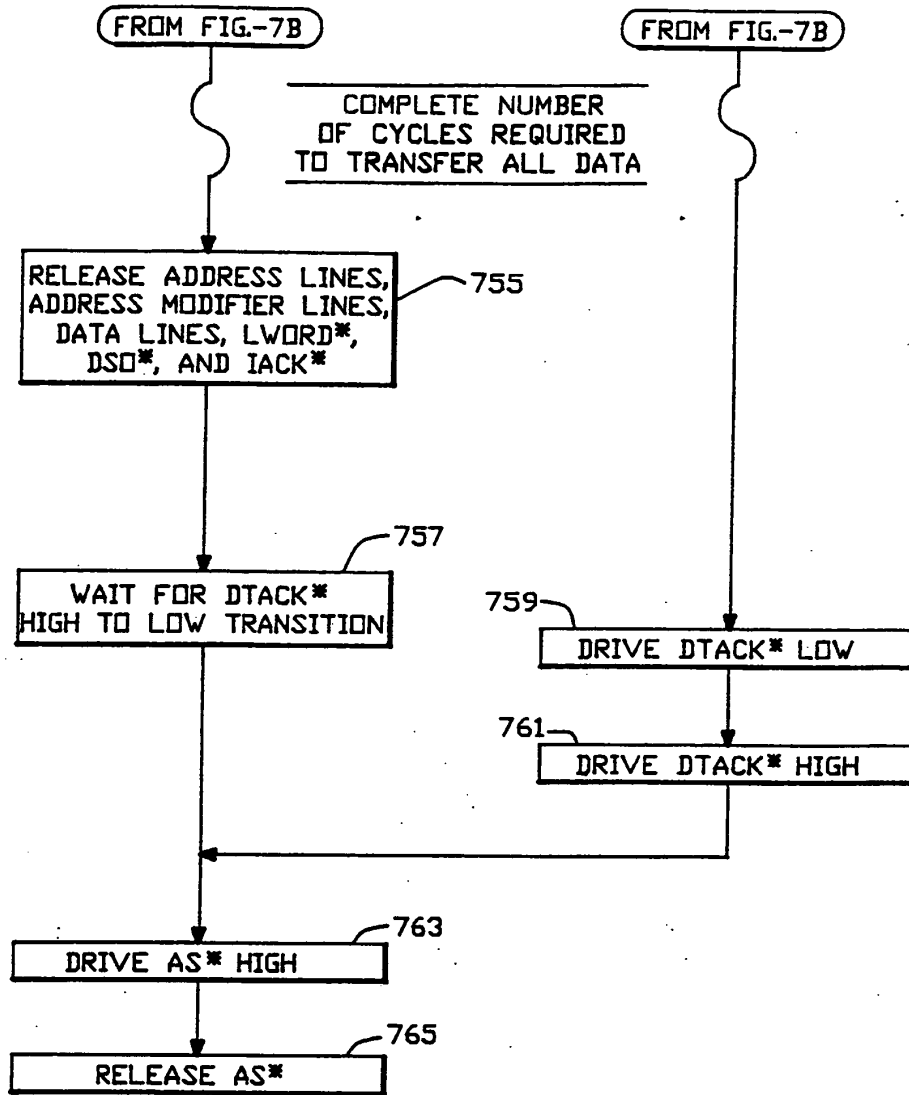


FIG.-7C

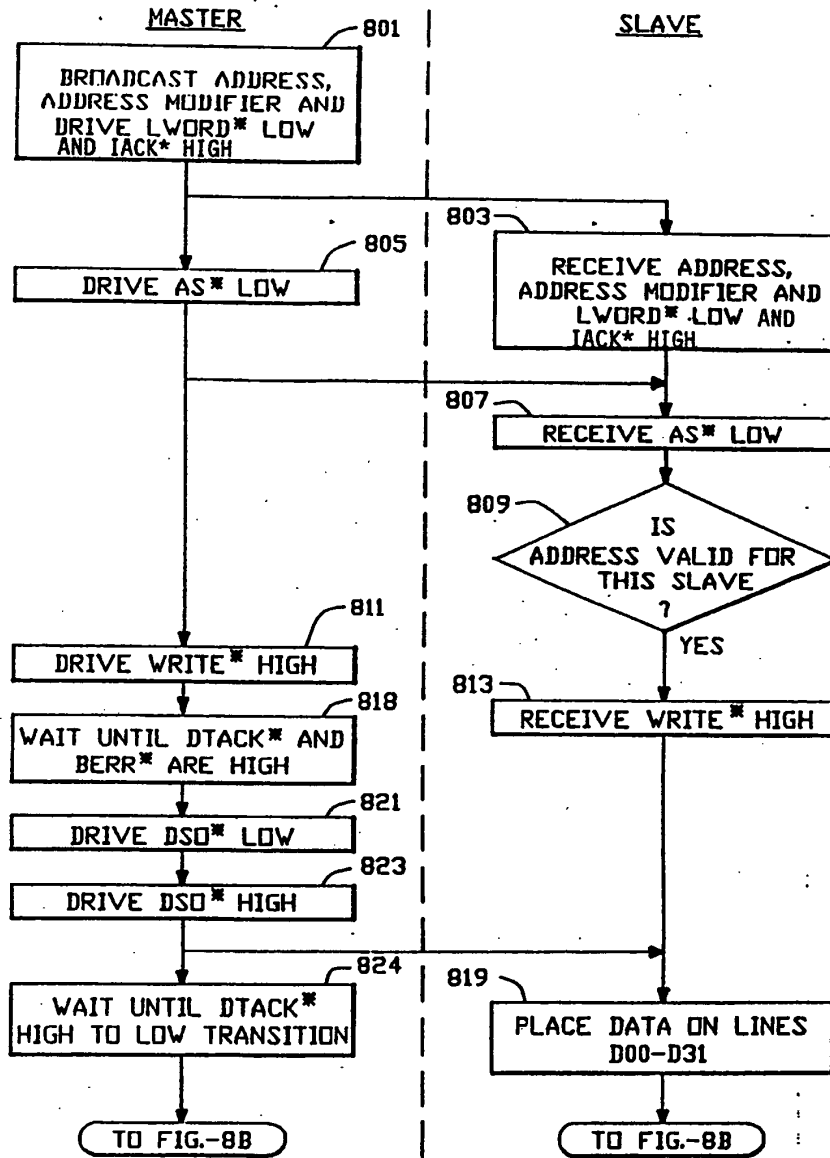


FIG.-8A

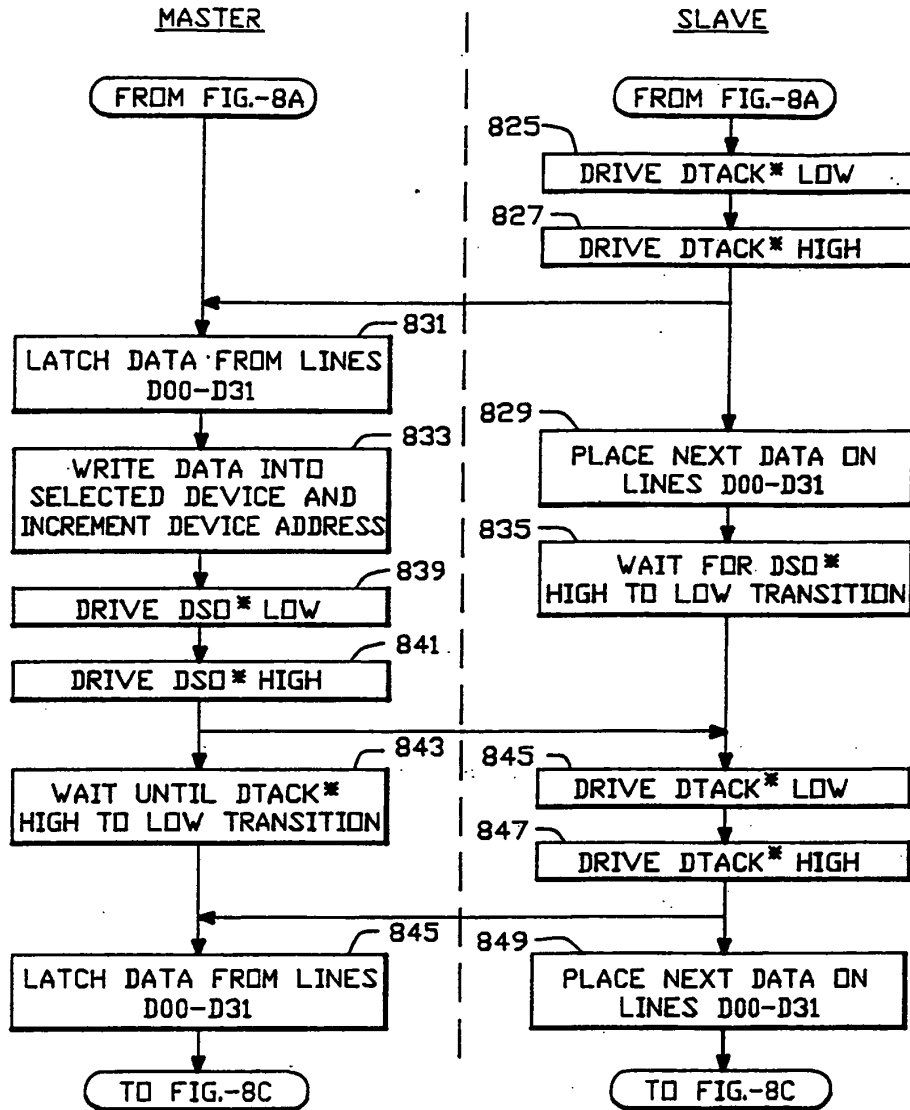


FIG.-8B

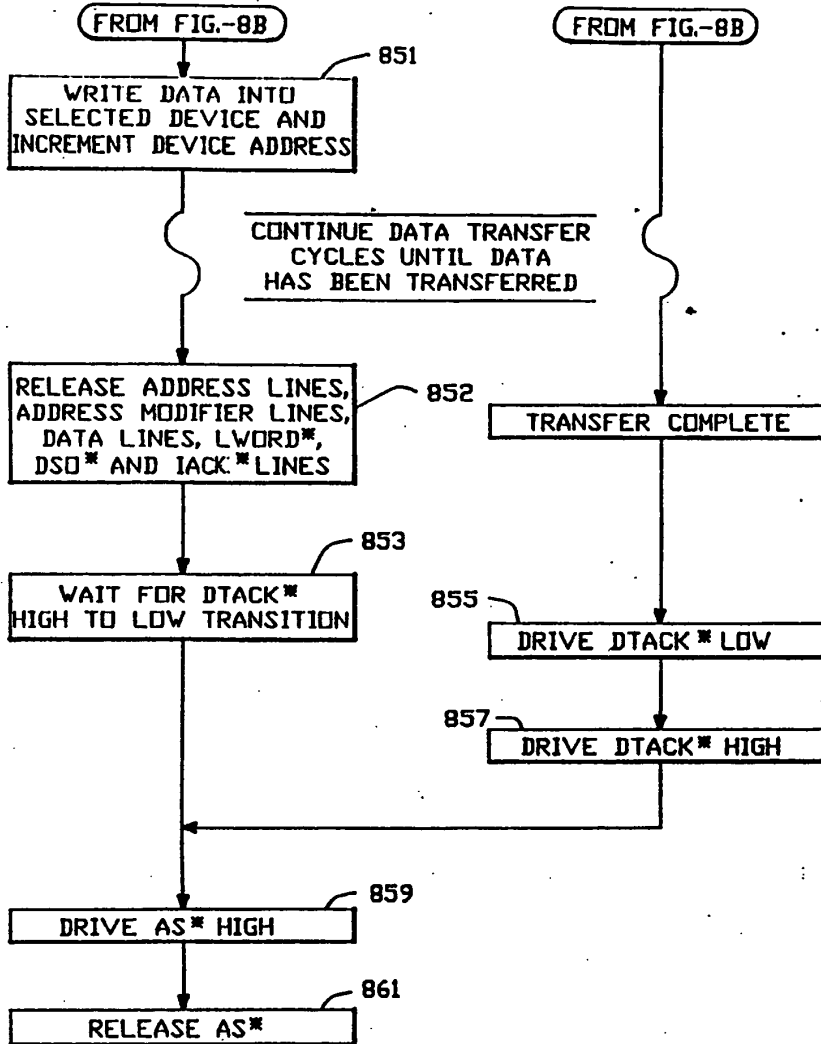


FIG.-8C

BKS  
R  
CCT  
R18

(19)  JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 10097493 A

(43) Date of publication of application: 14.04.98

(51) Int. Cl.	G06F 13/368 G06F 15/16	
(21) Application number:	09143702	(71) Applicant: SUN MICROSYST INC
(22) Date of filing:	02.06.97	(72) Inventor: SCHMAHL KENNETH A TEDONE MATTHEW J SCHELL JOHN C KARMINSKY IGOR CHAN RAY P
(30) Priority:	31.05.96 US 96 656641	

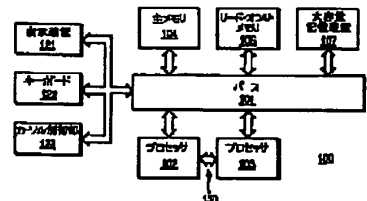
(54) METHOD AND DEVICE FOR GIVING BUS CONTROLLING RIGHT

COPYRIGHT: (C)1998,JPO

(57) Abstract:

**PROBLEM TO BE SOLVED:** To obtain a method for giving ownership of resources between plural devices without using an external arbiter by giving up an access to a bus when a timer terminates.

**SOLUTION:** A bus 101 is shared by two processors 102 and 103 in the same computer system 100. The processor 102 is allowed to access the bus 101 only while it is in its designated controlling right state and similarly, the processor 103 is allowed to access the bus 101 only while it is in its designated controlling right state. Thus, the scramble for the bus 101 is prevented by only either of the processors 102 and 103 accessing the bus 101 at a time. The bus controlling right state of the system is decided by a token, i.e., a signal which is generated by the processors 102 and 103. That is, the processors 102 and 103 generate a signal onto a line 130 each time they acquire an access to the bus 101, every time they abandon the access to the bus 101, or whenever they desire to acquire an access to the bus 101.



(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-97493

(43)公開日 平成10年(1998)4月14日

(51)Int.Cl. <sup>4</sup>	識別記号	F I	Z
G 0 6 F 13/368		G 0 6 F 13/368	
15/16	3 6 0	15/16	3 6 0 R

審査請求 未請求 請求項の数25 OL (全 10 頁)

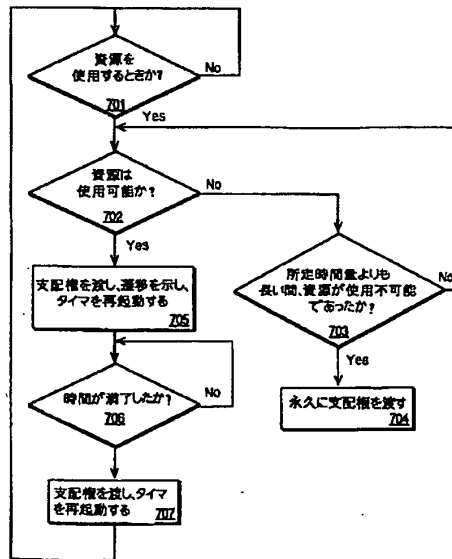
(21)出願番号	特願平9-143702	(71)出願人	597004720 サン・マイクロシステムズ・インコーポレ ーテッド Sun Microsystems, In c. アメリカ合衆国カリフォルニア州94043- 1100, マウンテン・ビュー, ガーシア・ア ヴェニュー 2550, エムエス・ピーエイ エル1-521
(22)出願日	平成9年(1997)6月2日	(74)代理人	弁理士 社本 一夫 (外4名)
(31)優先権主張番号	6 5 6 6 4 1		
(32)優先日	1996年5月31日		
(33)優先権主張国	米国 (US)		

最終頁に続く

(54)【発明の名称】 バス支配権を渡す方法および装置

(57)【要約】

【課題】 バスの支配権を渡す方法を提供する。  
【解決手段】 バスが使用可能か否かについて判定を行  
う(701)。バスが使用可能な場合(702)、バス  
にアクセスしかつ前記バスにアクセスしていることを示  
す信号を発生し、またバスへのアクセスに回答してタイ  
マを起動する(705)。タイマが満了したとき(70  
6)、前記バスへのアクセスを譲る(707)。





## 【特許請求の範囲】

【請求項1】バス支配権を渡す方法であって、バスが使用可能か否かについて判定を行うステップと、前記バスが使用可能な場合、前記バスにアクセスし、かつ前記バスにアクセスしていることを示す信号を発生するステップと、

前記バスへのアクセスに応答してタイマを起動するステップと、

前記タイマが満了したとき、前記バスへのアクセスを放棄するステップと、から成ることを特徴とするバス支配権渡し方法。

【請求項2】請求項1記載の方法であって、更に、前記バスへのアクセスを放棄した後前記タイマを再起動させるステップ、を含むことを特徴とするバス支配権渡し方法。

【請求項3】請求項1記載の方法であって、更に、前記バスへのアクセスを放棄したことを示す信号を発生するステップ、を含むことを特徴とするバス支配権渡し方法。

【請求項4】請求項1記載の方法であって、更に、前記バスが使用不可能な場合、所定時間量より長く前記バスがアクセスされているか否かについて判定を行い、かつ前記所定時間量より長く前記バスがアクセスされている場合、前記バスへのアクセスを獲得するステップ、を含むことを特徴とするバス支配権渡し方法。

【請求項5】請求項1記載の方法において、バスが使用可能か否かについて判定を行う前記のステップは、バス・エージェントがバスにアクセスしていることを示す信号を前記バス・エージェントが発生したか否かを調べるためにチェックを行うステップ、から成ることを特徴とするバス支配権渡し方法。

【請求項6】一連の命令を記憶してあるコンピュータ読み取り可能媒体であって、前記一連の命令は、プロセッサによって実行させたときに、

バスが使用可能か否かについて判定を行うステップと、前記バスが使用可能な場合、前記バスにアクセスし、かつ前記バスにアクセスしていることを示す信号を発生するステップと、

前記バスへのアクセスに応答してタイマを起動するステップと、

前記タイマが満了したとき、前記バスへのアクセスを放棄するステップと、を前記プロセッサに行わせること、を特徴とするコンピュータ読み取り可能記憶媒体。

【請求項7】請求項6記載のコンピュータ読み取り可能媒体であって、更に、前記プロセッサによって実行させたときに、前記バスへのアクセスを放棄した後前記タイマを再起動させるステップを、前記プロセッサに行わせる命令を含むこと、を特徴とするコンピュータ読み取り可能媒体。

【請求項8】請求項6記載のコンピュータ読み取り可能

媒体であって、更に、前記プロセッサによって実行させたときに、前記バスへのアクセスを放棄したことを示す信号を発生するステップを、前記プロセッサに行わせる命令を含むこと、を特徴とするコンピュータ読み取り可能媒体。

【請求項9】請求項6記載のコンピュータ読み取り可能媒体であって、更に、前記プロセッサによって実行させたときに、前記バスが使用不可能な場合、所定時間量より長く前記バスがアクセスされているか否かについて判定を行い、前記所定時間量より長く前記バスがアクセスされている場合、前記バスへのアクセスを獲得するステップを、前記プロセッサに行わせる命令を含むこと、を特徴とするコンピュータ読み取り可能媒体。

【請求項10】請求項6記載のコンピュータ読み取り可能媒体において、バスが使用可能か否かについて判定を行う前記のステップは、バス・エージェントが前記バスにアクセスしていることを示す信号を前記バス・エージェントが発生したか否かを調べるチェックを行うステップから成ること、を特徴とするコンピュータ読み取り可能媒体。

【請求項11】プロセッサであって、資源に結合した構成要素から第1信号を受信したとき、前記プロセッサに前記資源へのアクセスを許可し、前記構成要素から第2信号を受信したとき、前記資源へのアクセスを前記構成要素に譲る、資源アクセス・ユニット、を備えていることを特徴とするプロセッサ。

【請求項12】請求項11記載のプロセッサであって、更に、前記資源アクセス・ユニットに結合してあり、前記プロセッサが前記資源へのアクセスを獲得したときに第3の信号を発生し、前記プロセッサが前記資源へのアクセスを放棄したとき第4の信号を発生する信号発生ユニット、を備えていることを特徴とするプロセッサ。

【請求項13】請求項11記載のプロセッサであって、更に、前記信号発生ユニットに結合してあり、前記第3および第4の信号が発生されたときに、時間期間を割り当てるタイマを備えていることを特徴とするプロセッサ。

【請求項14】請求項11記載のプロセッサにおいて、前記構成要素は第2プロセッサであること、を特徴とするプロセッサ。

【請求項15】請求項11記載のプロセッサにおいて、前記構成要素は複数のプロセッサであること、を特徴とするプロセッサ。

【請求項16】請求項11記載のプロセッサにおいて、前記資源はバスであること、を特徴とするプロセッサ。

【請求項17】請求項11記載のプロセッサにおいて、前記資源はメモリであること、を特徴とするプロセッサ。

【請求項18】コンピュータ・システムであって、

A) バスと、  
 B) 前記バスに結合した第1プロセッサであって、  
 1) 前記第1プロセッサが前記バスへのアクセスを獲得したときに第1信号を発生し、前記第1プロセッサが前記バスへのアクセスを譲ったときに第2信号を発生する第1信号発生ユニットと、  
 2) 第3信号を受信したときに前記第1プロセッサに前記バスへのアクセスを許可し、第4信号を受信したときに前記バスへのアクセスを譲る第1バス・アクセス・ユニットと、を有する前記第1プロセッサと、  
 C) 前記バスおよび前記第1プロセッサに結合した第2プロセッサであって、  
 1) 前記第2プロセッサが前記バスへのアクセスを獲得したときに前記第4信号を発生し、前記第2プロセッサが前記バスへのアクセスを譲ったときに前記第3信号を発生する第2信号発生ユニットと、  
 2) 前記第2信号を受信したときに前記第2プロセッサに前記バスへのアクセスを許可し、前記第1信号を受信したときに前記バスへのアクセスを譲る第2バス・アクセス・ユニットと、を有する前記第2プロセッサと、から成るコンピュータ・システム。  
 【請求項19】請求項18記載のコンピュータ・システムであって、更に、前記第1および第2プロセッサに結合した記憶装置のアレイを備えていること、を特徴とするコンピュータ・システム。  
 【請求項20】請求項18記載のコンピュータ・システムであって、更に、前記バスに結合した環境サービス・センタを備えていること、を特徴とするコンピュータ・システム。  
 【請求項21】メモリと、バスと、表示装置とを含む、プロセッサをベースとするシステムと通信するように構成したバス・エージェント内に常駐するバス仲裁装置であって、前記バスに結合した構成要素からの第1信号を受信したときに、前記バス・エージェントに前記バスへのアクセスを許可し、前記構成要素からの第2信号を受信したときに、前記バスへのアクセスを前記構成要素に譲る、資源アクセス・ユニット、を備えていることを特徴とするバス仲裁装置。  
 【請求項22】請求項21記載のバス仲裁装置であって、更に、前記資源アクセス・ユニットに結合してあり、前記バス・エージェントが前記資源へのアクセスを獲得したときに第3信号を発生し、前記バス・エージェントが前記資源へのアクセスを譲ったときに第4信号を発生する信号発生ユニット、を備えていることを特徴とするバス仲裁装置。  
 【請求項23】第1バス・エージェントと第2バス・エージェントとの間でバスを仲裁するシステムであって、前記第1バス・エージェントが前記バスへのアクセスを

獲得したときに第1信号を発生し、前記第1バス・エージェントが前記バスへのアクセスを譲ったときに第2信号を発生する第1信号発生ユニットと、  
 第3信号を受信したときに前記第1バス・エージェントに前記バスへのアクセスを許可し、第4信号を受信したときに前記バスへのアクセスを譲る第1バス・アクセス・ユニットであって、前記第1信号発生ユニットと前記第1バス・アクセス・ユニットは前記第1バス・エージェント内に常駐した、前記の第1バス・アクセス・ユニットと、  
 前記第2バス・エージェントが前記バスへのアクセスを獲得したときに前記第4信号を発生し、前記第2バス・エージェントが前記バスへのアクセスを譲ったときに前記第3信号を発生する第2信号発生ユニットと、  
 前記第2信号を受信したときに前記第2バス・エージェントに前記バスへのアクセスを許可し、前記第1信号を受信したときに前記バスへのアクセスを譲る第2バス・アクセス・ユニットであって、前記第2信号発生ユニットと前記第2バス・アクセス・ユニットは前記第2バス・エージェント内に常駐した、前記の第2バス・アクセス・ユニットと、から成るバス仲裁システム。  
 【請求項24】請求項23記載のシステムであって、更に、前記第1および第2バス・エージェントに結合した記憶装置のアレイを備えていること、を特徴とするバス仲裁システム。  
 【請求項25】請求項23記載のシステムであって、更に、前記バスに結合した環境サービス・センタを備えていること、を特徴とするバス仲裁システム。  
 【発明の詳細な説明】  
 【0001】  
 【発明の属する技術分野】本発明は、バス調整 (bus regulation) の分野に関する。更に特定すれば、本発明は、多数の装置間でバス支配権を渡す方法および装置に関するものである。  
 【0002】  
 【従来の技術】多数の装置がバス上に常駐する場合、このバスに対するアクセスの調整が必要となる。バスに対するアクセスを調整することによって、通信を望む多数の装置が同時に異なる転送のために制御ラインおよびデータ・ラインをアサートしてバス競合が発生しないことを保証する。  
 【0003】バス・アクセスを調整する手法の1つに、システムにおいて1つ以上のバス・マスタを使用することが挙げられる。バス・マスタとは、バスに対するアクセスを制御するものである。これは、全てのバス要求を開始 (initiate) し制御する。プロセッサはメモリ装置に対するアクセスについてのバス要求を開始できなければならないので、常に1つのバス・マスタとなる。メモリ装置はリード (read) およびライト (write) 要求に応答するが、決してそれ自体の要求を発生しないので、通

常スレーブである。

【0004】多数の中央演算ユニット(CPU)がある場合、または入出力(I/O)装置がバス・トランザクションを開始できる場合、バスは多数のマスタを有することになる。多数のマスタがある場合、どれが次にバスを獲得するかを決定するために、マスタ間の仲裁法(arbitration scheme)が必要となる。この仲裁法を実施するにはバス・アービタを典型的に用いる。あるバス仲裁法においては、バスの使用を望む装置がバス要求を送り、その後バスをこれに付与する。付与後、その装置はバスを使用することができ、後に、バスをもはや必要としないことをバス・アービタに知らせる。すると、バス・アービタは別の装置にバスを付与することができる。殆どどのマルチ・マスタ・バス(multi-master bus)は、要求および付与を行うために1組のバス信号を有している。各装置がバスを解放するためにそれぞれ自体の要求ラインを用いない場合、バス解放ラインも必要となる。ときとして、バス仲裁に用いるそれら信号が物理的に個別のラインを有することがあり、一方他のシステムでは、バスのデータ・ラインをこの機能のために用いるものがある。仲裁は多くの場合固定の優先順位であり、デジー・チェーン装置の場合と同様である。あるいは、どのマスタがバスを獲得するかを不規則に選択するほぼ公正な方法がある。

【0005】

【発明が解決しようとする課題】しかしながら、バス・アービタの使用にはいくつかの欠点がある。バス・アービタを追加することにより、動作のために追加の電力が必要となる。これは、厳しい電力制限の下で動作するコンピュータ・システムにとっては問題である。また、バス・アービタを実装すると、コンピュータ・システム内に追加の空間も必要となる。このように、コンピュータ・システムの環境によっては、物理的空間の可用性がバス・アービタの実装を許さない場合がある。恐らく最も重要なのは、仲裁の目的のために追加の構成要素を用いることによって、コンピュータ・システム全体に望ましくないコストを付加することである。

【0006】したがって、必要としているのは、外部のアービタを用いることなく、複数の装置間で資源の所有権を渡す装置である。

【0007】

【課題を解決するための手段】以下、資源の支配権を渡す方法について説明する。この方法によれば、バスを使用すべきか否かを判定する。バスを使用すべき場合、バスが使用可能か否かの判定を行う。バスが使用可能な場合、バスにアクセスし、バスをアクセスしていることを示す信号を発生する。また、タイマも起動し、このタイマが満了したときバスへのアクセスを譲る。

【0008】次に、共有資源の支配権を渡すプロセッサについて説明する。このプロセッサは資源アクセス・ユ

ニットを備えている。資源アクセス・ユニットは、資源に結合してある構成要素からの第1信号を受信したとき、そのプロセッサの資源へのアクセスを許可する。資源アクセス・ユニットは、前記構成要素からの第2信号を受信したとき、資源へのアクセスを前記構成要素に譲る。更に、プロセッサは信号発生ユニットを備えている。この信号発生ユニットは資源アクセス・ユニットに結合してある。信号発生ユニットは、プロセッサが資源へのアクセスを獲得したときに第3信号を発生し、プロセッサが資源へのアクセスを譲ったときに第4信号を発生する。

【0009】

【発明の実施の形態】添付図面においては、本発明を限定としてではなく一例として示す。尚、同様の参照番号は同様の要素を示すものとする。

【0010】これより、メモリ内のデータにアクセスする方法および装置について説明する。以下の記述においては、説明の目的のために、多数の具体的な詳細を示し、本発明の完全な理解が得られるようにしてある。しかしながら、本発明はこれら具体的な詳細がなくても実現可能であることは、当業者には明白であろう。他の場合には、公知の構造および装置をブロック図形式で示すことにより、本発明を不必要に曖昧にするのを回避する。

【0011】図1を参照すると、本発明の好適実施例を実装可能なコンピュータ・システムを100で示している。コンピュータ・システム100は、情報を通信するためのバスまたはその他の通信手段101、およびバス101に結合してあり情報を処理するプロセッサ102、103を備えている。更に、システム100は、バス101に結合してあり、情報およびプロセッサ102、103が実行する命令を格納する、ランダム・アクセス・メモリ(RAM)またはその他のダイナミック記憶装置104(主メモリと呼ぶ)も備えている。主メモリ104は、プロセッサ102、103による命令の実行中に、一時的な変数やその他の中間情報を格納するためにも使用可能である。また、コンピュータ・システム100は、バス101に結合してあり、プロセッサ102、103のために静的な情報および命令を格納するリード・オンリ・メモリ(ROM)および/またはその他のスタティック記憶装置106も備えている。データ記憶装置107はバス101に結合してあり、情報および命令を格納する。プロセッサ102または103によって実行可能なコンピュータ読み取り可能媒体からの命令は、データ記憶装置107に格納することができる。磁気ディスクまたは光ディスクのようなデータ記憶装置107およびそれに対応するディスク駆動装置をコンピュータ・システム100に結合することができる。

【0012】また、コンピュータ・システム100は、コンピュータ・ユーザに情報を表示するために、バス1

01を通じて、陰極線管(CRT)のような表示装置121にも結合可能である。英数字およびその他のキーを含む英数字入力装置122が、通常、バス101に結合しており、情報およびコマンド選択をプロセッサ102、103に伝達する。別のタイプのユーザ入力装置には、マウス、トラックボール、または方向情報およびコマンド選択をプロセッサ102に伝達すると共に、表示装置121上でのカーソルの動きを制御するカーソル方向キーのようなカーソル制御部123がある。この入力装置は、典型的に、2方向の軸、即ち第1軸(例えば、x)および第2軸(例えば、y)に2の自由度を有し、入力装置が平面内における位置を指定できるようになっている。

【0013】あるいは、スタイラスやペンのような他の入力装置を用いて表示装置との双方向通信を行うことも可能である。スタイラスまたはペンを用いてコンピュータ画面上に表示させた物に触れることによって、表示物を選択することができる。コンピュータは、接触感応画面を実装することによってその選択を検出する。同様に、ライト・ペンおよび光感応画面を用いて、表示物を選択することも可能である。したがって、このような装置は、マウスまたはトラックボールを内蔵したシステムにおけるような「ポイントおよびクリック(point and click)」の代わりに、単一の動作として、選択位置および選択を検出することができる。スタイラスおよびペンに基づく入力装置、ならびに接触および光感応画面は、当該分野においては公知である。このようなシステムは、122のようなキーボードがなくともよく、その場合、全てのインターフェースは、スタイラスを通じた書き込み器具(ペンのような)として提供し、書き込んだテキストは光学文字認識(OCR)技術を用いて解釈する。

【0014】図1は本発明の一実施例を示し、バス101は、同一コンピュータ・システム100内の2つのプロセッサ102、103間で共有する。バス競合を防止するためには、一度にプロセッサ102または103の一方のみがバス101にアクセスするようにすればよい。プロセッサ102には、それに指定したバス支配権状態の間のみ、バス101へのアクセスを許可する。同様に、プロセッサ103には、それに指定したバス支配権状態の間のみ、バス101へのアクセスを許可する。システムのバス支配権状態は、プロセッサ102、103が発生するトークン即ち信号によって決定する。本発明の一実施例では、プロセッサ102、103は、バス101へのアクセスを獲得する毎、バス101へのアクセスを放棄する毎、またはバス101へのアクセスの獲得を希望する毎に、ライン130上に信号を発生する。本発明の別の実施例では、プロセッサの一方がライン130上に発生する信号は、単一の信号でも複数の信号でもよい。プロセッサ102が発生する信号は、ライン1

30を通じてプロセッサ103に送り、プロセッサ103が発生する信号はライン130を通じてプロセッサ102に送る。各プロセッサには、それ自体が発生した信号および他方のプロセッサが発生した信号のコピーをもたせる。各プロセッサは、システム100の現在のバス支配権状態を認識している。

【0015】図2は本発明の一実施例を示し、ここでは、第1コンピュータ・システム250からのプロセッサ102および第2コンピュータ・システム251からの第2のプロセッサ202が、共有資源210に対するアクセスを共有する。共有資源210は、一度にプロセッサ102またはプロセッサ202のいずれか一方によってのみアクセスが可能な資源である。共有資源210は、例えば、バスまたはメモリである。共有資源210は直接プロセッサ102、202に結合するか、あるいは他のバスまたは構成要素を介してプロセッサ102、202に結合することもできる。プロセッサ102には、それに指定した資源支配権状態の間のみ、共有資源210へのアクセスを許可する。プロセッサ202には、それに指定した資源支配権状態の間のみ、共有資源210へのアクセスを許可する。システムの資源支配権状態は、プロセッサ102、202が発生するトークン即ち信号によって決定する。本発明の一実施例では、プロセッサ102、202は、これらが共有資源210へのアクセスを獲得する毎、共有資源210へのアクセスの獲得を希望する毎に、信号を発生する。本発明の一実施例では、プロセッサ102または202が発生する信号は、単一の信号でも複数の信号でもよい。プロセッサ102が発生した信号はライン230を通じてプロセッサ202に送り、プロセッサ202が発生した信号はライン230を通じてプロセッサ102に送る。各プロセッサは、それ自体および他方のプロセッサが発生した信号のコピーを有する。各プロセッサは、コンピュータ・システムの現在のバス支配権状態について認識している。

【0016】図3は、大容量記憶システム300に実施した場合の本発明の実施例を示す。大容量記憶システム300は、ハード・ディスク構造体(HDA: hard disk assembly)331に結合した第1記憶要素アレイ335、およびハード・ディスク構造体(HDA)341に結合した第2記憶要素アレイ345を備えている。第1および第2記憶要素アレイ335、345は、ホスト・インターフェース・ユニット304または314の一方およびバス301または311の一方を通じて、ホスト(図示せず)によるアクセスを受ける。バス301、311は、例えば、従来のファイバ・チャネル・インターフェース、直列記憶アーキテクチャ・インターフェース、小型コンピュータ・システム・インターフェース(SCSI)、P1394インターフェース、またはその他の公知のインターフェースによって実施可能であ

る。ハード・ディスク構造体331は、バス301を用いて、第1記憶要素アレィ335にインターフェースするように動作する。ハード・ディスク構造体331は、プロセッサ302、312が読み取るデータを格納するために使用するレジスタ332を含む。ハード・ディスク構造体341は、バス311を用いて、第2記憶要素アレィ345にインターフェースするように動作する。ハード・ディスク構造体341は、プロセッサ302、312が読み取るデータを記憶するために使用するレジスタ342を含む。

【0017】環境サービス・センタ(ESC: environmental service center)325は、温度制御や大容量記憶システム300への給電のような環境サービスを提供する。また、環境サービス・センタ325は、大容量記憶システム300の環境サービスに関するデータも提供する。環境サービス・センタ325は、任意の公知の回路によって実現すればよい。プロセッサ302は、バス301および共有バス320に結合してある。プロセッサ302は、共有バス320を通じて環境サービス・センタ325から環境サービス・データを読み取ることによって、環境サービス・センタ325をポーリングする。プロセッサ302は、環境サービス・データをメモリ・ユニット303に格納する。プロセッサ302は、大容量記憶システム300の環境を監視するよう動作し、環境が許容範囲を外れた場合にシステムの健全性を維持する。同様に、プロセッサ312は、共有バス320を通じて環境サービス・センタ325から環境サービス・データを読み出すことによって、環境サービス・センタ325をポーリングする。プロセッサ312は環境サービス・データをメモリ・ユニット313に格納する。プロセッサ312は、大容量記憶システム300の環境を監視するよう動作し、環境が許容範囲を外れた場合に、システムの健全性を維持する。

【0018】環境サービス・センタ325からの環境サービス・データは、一度にプロセッサ302、312の一方しか共有バス320を通じてアクセスすることができない。プロセッサ302は、それに指定されたバス支配権状態の間のみ共有バス320へのアクセスを許可される。システム300のバス支配権状態は、プロセッサ302、312が発生するトークン即ち信号によって決定する。本発明の一実施例では、プロセッサの一方がバス320へのアクセスを獲得したとき、バス320へのアクセスを放棄したとき、またはバス320へのアクセスの獲得を希望したときに、プロセッサ302または312が発生する信号によって、バス支配権状態を変更する。本発明の別の実施例では、各プロセッサ302または312が発生する信号は、単一の信号または複数の信号でもよい。本発明の更に別の実施例では、共有バス320の支配権を新たなマスタが獲得する毎に、プロセッサ302内のタイマ355およびプロセッサ312内

のタイマ356をセットする。タイマ355、356がタイムアウトする毎に、共有バス320の支配権が移る。プロセッサ302が発生する信号は、ライン350を通じてプロセッサ312に送り、プロセッサ312が発生する信号はライン350を通じてプロセッサ302に送る。各プロセッサは、それ自体および他方のプロセッサが発生した信号のコピーを有する。各プロセッサ302、312は、システム300の現在のバス支配権状態について認識している。

10 【0019】本発明の一実施例では、システム300のプロセッサ302、312が認識するバス支配権状態には4種類ある。図4はこの4種類の状態を示す表である。状態1では、プロセッサ302(装置1)が共有バス320の支配権を有する。状態1が発生するのは、プロセッサ302が0信号をライン350上に発生し、プロセッサ312(装置2)が0信号をライン350上に発生するときである。状態2では、バス支配権をプロセッサ302からプロセッサ312に移転させる。状態2が発生するのは、プロセッサ302が1信号をライン350上に発生し、プロセッサ312が0信号をライン350上に発生するときである。状態3では、プロセッサ312が共有バス320の支配権を有する。状態3が発生するのは、プロセッサ302が1信号をライン350上で発生し、プロセッサ312が1信号をライン350上で発生するときである。状態4では、バス支配権をプロセッサ312からプロセッサ302に移転させる。状態4が発生するのは、プロセッサ302が0信号をライン350上に発生し、プロセッサ312が1信号をライン350上に発生するときである。図5は、図4に示した状態1ないし4を実行する順序を示す状態図である。状態の数、これらの状態を実行する順序、および状態を表わすために用いる信号の数は、本発明の実装に応じて変更してもよいことは認められよう。

【0020】図6は、プロセッサ302の一実施例を示す。プロセッサ302は計算および制御ユニット610を含む。本発明の一実施例では、計算および制御ユニット610は、2つのファイバ・チャンネル仲裁式ループ・ポート(fiber channel arbitrated loop port)、1プロックの埋め込みRAM、ホスト・バス・インターフェース、および処理ユニットを含む。計算および制御ユニット610は、環境サービス・センタからの環境サービス・データをポーリングし、コンピュータ・システム300の環境を制御するように動作する。

【0021】更に、プロセッサ302は、資源アクセス・ユニット620、タイマ355、および信号発生ユニット630を含む。資源アクセス・ユニット620は、メモリ記憶システム300のバス支配権状態を追跡し、プロセッサ302が共有バス320の支配権を受けたときに、計算および制御ユニット610に環境サービス・センタ325をポーリングするように制御する。資源ア

クセス・ユニット620はライン350を通じてプロセッサ312からの信号を受信する。この信号は、いつプロセッサ320が次の状態に移る準備が完了しているかを示す。資源アクセス・ユニット620はタイマ355に結合してある。資源アクセス・ユニット620は、新たなマスタがバス320の支配権を取ったときに、タイマ355をリセットする。所定の時間量の後、タイマ355はタイムアウトする。これは、共有バス320が他のマスタに渡されることを、資源アクセス・ユニット620に通知するものである。資源アクセス・ユニット620は、信号発生ユニット630に命令して、ライン631上に信号を発生させ、プロセッサ302が次の状態に移る準備ができたことを示すようにさせる。システム300のバス支配権状態は、プロセッサ302、312が発生する信号によって決定する。資源アクセス・ユニット620、タイマ355および信号発生ユニット630は、ハードウェア、ソフトウェア、またはハードウェアおよびソフトウェアの組み合わせによって実施可能である。図6に示す本発明の実施例では、資源アクセス・ユニット620、タイマ355、および信号発生ユニット630は、計算および制御ユニット610外部のハードウェアとして実施してある。本発明の代替実施例では、資源アクセス・ユニット620および信号発生ユニット630は、プロセッサ302が実行する命令の集合によって実現するソフトウェア・モジュールである。プロセッサ312は、プロセッサ302と同様に動作し、プロセッサ302を実現する際に使用可能なものと同一の構成要素によって実現すればよい。

【0022】本発明は、外部のアービタを使用することなく、いずれも他方のマスタではない2つの装置間において、共有資源に対する支配権の仲裁を可能にするものである。資源アクセス・ユニットおよび信号発生ユニットをソフトウェアで実現した本発明の好適実施例では、システムからの電力や空間の追加を必要とせずに、仲裁を達成する。

【0023】図6は、資源アクセス・ユニット620、信号発生ユニット630およびタイマ355がプロセッサ302内に常駐する本発明の実施例を示すが、これらの構成要素は、共有資源へのアクセスを共有し、共有資源へのアクセスを仲裁するいずれかのエージェント(agent)に常駐させてもよいことは認められよう。

【0024】本発明の一実施例では、プロセッサ32が環境サービス・センタ302からの環境サービス・データをポーリングした後で、かつプロセッサ302が共有バス320のバス支配権を有する状態にシステム300がある間に、プロセッサ302は主メモリ313内の環境サービス・データを更新する。本発明のこの実施例では、プロセッサ312が環境サービス・センタ325からの環境サービス・データをポーリングした後で、かつプロセッサ312が共有バス320のバス支配権を有す

る状態にシステム300がある間にも、プロセッサ312は主メモリ303内の環境サービス・データを更新する。

【0025】プロセッサ302は、データ交換によって、主メモリ313内の環境サービス・データを更新する。ライン350がデータ・ポーリングの間に共有バス320の支配権について通信する場合と同様に、データ交換の間、プロセッサ302、312間の共有バス320の支配権について通信するために、第2ライン(図示せず)を用いる。プロセッサ302は、データ交換の間それが共有バス320の支配権を有する場合、ハード・ディスク構造体332、342のレジスタ332、342に環境サービス・データを書き込む。プロセッサ312は、データ交換の間それが共有バス320の支配権を有する場合、レジスタ332、342から環境システム・データを読み出し、このデータをメモリ・ユニット313に格納する。プロセッサ302は、メモリ・ユニット303内の全ての環境サービス・データをレジスタ332、342に書き込み、そして主メモリ313に転送し終わるまで、新たなデータをレジスタ332、342に書き込み続ける。プロセッサ312が共有バス320の支配権を有する状態にシステム300があるとき、プロセッサ312は、メモリ・ユニット303内の環境サービス・データを更新する際のプロセッサ302と同様に動作する。本発明の代替実施例では、プロセッサ302、312は単一ラインおよび単一の集合の信号を用いて、環境サービス・データのポーリングおよび交換の間に、共有バス320の支配権を渡す。

【0026】プロセッサ302が動作不能となり、所定時間期間内にそれが次のバス支配権状態に移る準備ができていない状況では、プロセッサ312内のタイマがタイムアウトする。これによって、プロセッサ302が動作不能であることをプロセッサ312に示す。これにตอบสนองして、プロセッサ312は共有バス320の排他的バス支配権を取る。同様に、プロセッサ312が動作不能となり、所定時間期間内にそれが次のバス支配権状態に移る準備ができていない状況では、プロセッサ302内のタイマがタイムアウトする。これによって、プロセッサ312が動作不能であることをプロセッサ302に示す。これにตอบสนองして、プロセッサ302は共有バス320の排他的バス支配権を取る。

【0027】図7は、2つの装置間で共有資源の支配権を渡す方法を示すフローチャートである。ステップ701において、共有資源を使用するか否かについて判定を行う。この判定を行うには、第1装置が資源へのアクセスを得てからの時間を記録しているタイマをチェックする。第1所定時間量の後、タイマがタイムアウトして、第2装置が共有資源にアクセスするときであることを示

す。共有資源を使用するときではない場合、制御はステップ701に戻る。共有資源を使用するときになった場合、制御はステップ702に進む。

【0028】ステップ702において、共有資源が使用可能か否かについて判定を行う。この判定は、資源アクセス・ユニットをチェックして現在の資源支配権状態を確かめることにより行える。資源支配権状態が、第1装置が支配権を有する状態である場合、共有資源は使用不可能であり、制御はステップ703に移る。共有資源が使用可能な場合、制御はステップ705に進む。

【0029】ステップ703において、第1装置が第2所定時間量にわたって、共有資源の支配権を有しているか否かについて判定を行う。この判定を行うには、第1装置が共有資源へのアクセスを得たときの時刻を記録しているタイマをチェックする。第1装置が共有資源の支配権を第2所定時間以上にわたって有していなかった場合、制御はステップ702に戻る。第1装置が第2所定時間量以上にわたって共有資源の支配権を有していた場合、制御はステップ704に進む。

【0030】ステップ704において、第2装置に共有資源の排他的支配権を与え、そして共有資源のそれ以降のマスタ候補として考慮することから、第1装置を除外する。

【0031】ステップ705において、第2装置に共有資源の支配権を与える。第2装置が共有資源にアクセスしたことを示す信号を発生し、そしてタイマをリセットする。

【0032】ステップ706において、共有資源の支配権を異なった装置に渡すべきか否かについて判定を行う。この判定を行うには、タイマが第1所定時間期間を過ぎてタイムアウトしたか否かを調べるチェックを行えばよい。タイマが第1所定時間期間を越えてタイムアウトしている場合、異なった装置に共有資源の支配権を渡すときであり、制御はステップ707に進む。タイマが第1所定時間期間を越えておらず、未だタイムアウトしていない場合、制御はステップ706に戻る。

【0033】ステップ707において、第2装置が信号を発生して、第2装置が共有資源のマスタではない次の資源支配権の状態へと、第2装置が遷移する準備ができていることを示す。制御はステップ701に進む。

【0034】以上の説明では、本発明についてその具体的な実施例を参照しながら記述した。しかしながら、本発明のより広範な主旨および範囲から逸脱することなく、

様々な変更や変化も可能であることは明白である。したがって、本明細書および図面は、限定的な意味ではなく例示的な意味で解釈すべきである。

【図面の簡単な説明】

【図1】本発明の一実施例を実装したマルチ・プロセッサ・コンピュータ・システムを示す図。

【図2】本発明の一実施例を実装した異なる2台のコンピュータ・システムからのプロセッサを示す図。

【図3】大容量記憶システムに実施した場合の本発明を示す図。

【図4】本発明の一実施例における支配権状態を示す表。

【図5】図4に示した状態の遷移順序を示す状態図。

【図6】本発明を実施するプロセッサの一実施例を示すブロック図。

【図7】共有資源の支配権を渡す方法を示すフロー・チャート。

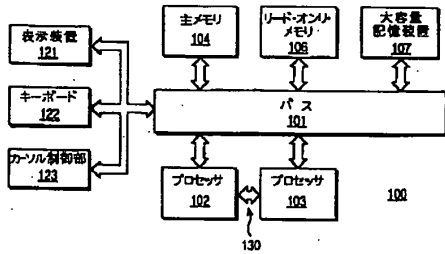
【符号の説明】

- 100 コンピュータ・システム
- 101 バス
- 102, 103 プロセッサ
- 130 ライン
- 202 第2プロセッサ
- 210 共有資源
- 230 ライン
- 250 第1コンピュータ・システム
- 251 第2コンピュータ・システム
- 300 大容量記憶システム
- 301, 311 バス
- 302, 312 プロセッサ
- 304, 314 ホスト・インターフェース・ユニット
- 303, 313 メモリ・ユニット
- 320 共有バス
- 325 環境サービス・センタ
- 331, 341 ハード・ディスク構造体
- 332, 342 レジスタ
- 335 第1記憶要素アレイ
- 345 第2記憶要素アレイ
- 355, 356 タイマ
- 610 計算および制御ユニット
- 620 資源アクセス・ユニット
- 630 信号発生ユニット

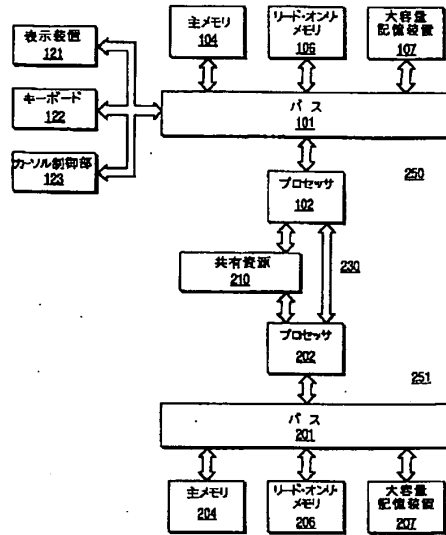
【図4】

状態	装置1	装置2	支配権
1	0	0	装置1がマスタである
2	1	0	支配権を装置1から装置2に渡す
3	1	1	装置2がマスタである
4	0	1	支配権を装置2から装置1に渡す

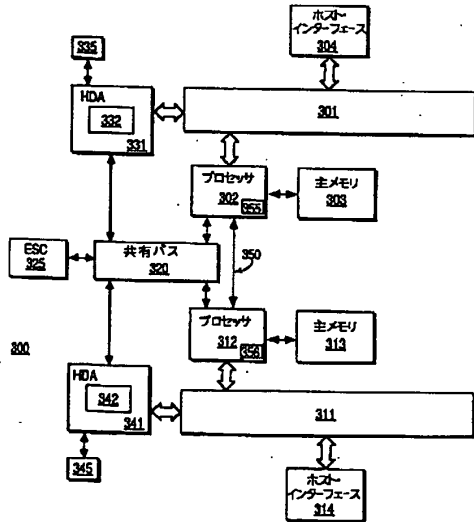
【図1】



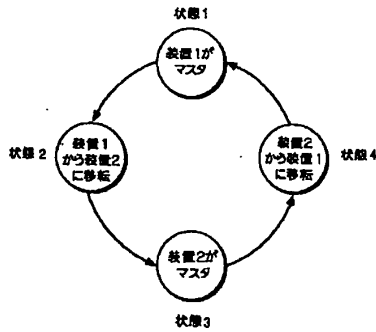
【図2】



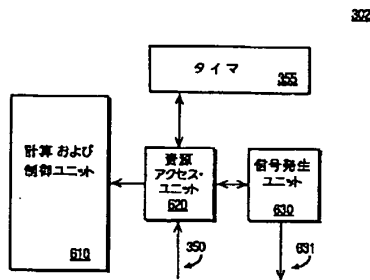
【図3】



【図5】

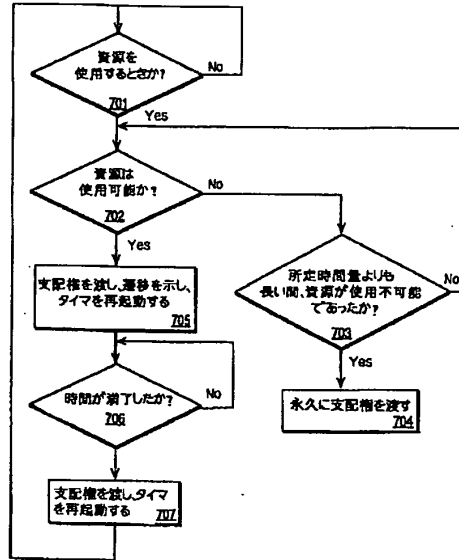


【図6】





【図7】



フロントページの続き

(71)出願人 597004720  
 2550 Garcia Avenue, MS  
 PAL1-521, Mountain V  
 iew, California 94043-  
 1100, United States of  
 America  
 (72)発明者 ケネス・エイ・シュマール  
 アメリカ合衆国カリフォルニア州95136,  
 サン・ホセ, ラッセンパーク・サークル  
 321

(72)発明者 マシュー・ジェイ・テダン  
 アメリカ合衆国カリフォルニア州94087,  
 サニーヴェイル, レミントン・ドライブ  
 887  
 (72)発明者 ジョン・シー・スケル  
 アメリカ合衆国カリフォルニア州94087,  
 サニーヴェイル, ファルコン・アヴェニュー  
 1428  
 (72)発明者 イゴール・カーミンスキー  
 アメリカ合衆国カリフォルニア州95129,  
 サン・ホセ, バイン・グローヴ・ウェイ  
 1480  
 (72)発明者 レイ・ビー・チャン  
 アメリカ合衆国カリフォルニア州95014,  
 カッパーティノ, プレイサー・スプリング  
 ス・コート 11851

JP1997251437A

1997-9-22

B2B  
19

**Bibliographic Fields**

**Document Identity**

(19)【発行国】

日本国特許庁(JP)

(12)【公報種別】

公開特許公報(A)

(11)【公開番号】

特開平9-251437

(43)【公開日】

平成9年(1997)9月22日

**Public Availability**

(43)【公開日】

平成9年(1997)9月22日

**Technical**

(54)【発明の名称】

計算機装置及び連続データサーバ装置

(51)【国際特許分類第6版】

G06F 13/372

13/00 353

13/16 520

【F1】

G06F 13/372 C

13/00 353 N

13/16 520 B

【請求項の数】

15

【出願形態】

OL

【全頁数】

18

**Filing**

【審査請求】

未請求

(21)【出願番号】

(19) [Publication Office]

Japan Patent Office (JP)

(12) [Kind of Document]

Unexamined Patent Publication (A)

(11) [Publication Number of Unexamined Application]

Japan Unexamined Patent Publication Hei 9- 251437

(43) [Publication Date of Unexamined Application]

1997 (1997) September 22\*

(43) [Publication Date of Unexamined Application]

1997 (1997) September 22\*

(54) [Title of Invention]

COMPUTER DEVICE AND CONTINUAL DATA  
SERVER DEVICE

(51) [International Patent Classification, 6th Edition]

G06F13/372

13/00353

13/16520

[F1]

G06F13/372C

13/00353N

13/16520B

[Number of Claims]

15

[Form of Application]

OL

[Number of Pages in Document]

18

[Request for Examination]

Unrequested

(21) [Application Number]

JP1997251437A

1997-9-22

特願平8-61467

Japan Patent Application Hei 8- 61467

(22)【出願日】

(22) [Application Date]

平成8年(1996)3月18日

1996 (1996) March 18\*

**Parties**

**Applicants**

(71)【出願人】

(71) [Applicant]

【識別番号】

[Identification Number]

000003078

000003078

【氏名又は名称】

[Name]

株式会社東芝

**TOSHIBA CORPORATION (DB 69-054-3517)**

【住所又は居所】

[Address]

神奈川県川崎市幸区堀川町72番地

Kanagawa Prefecture Kawasaki City \*\*Horikawa-cho 72

**Inventors**

(72)【発明者】

(72) [Inventor]

【氏名】

[Name]

浅野 滋博

Asano \*\*

【住所又は居所】

[Address]

神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

Kanagawa Prefecture Kawasaki City \*\*Komukai Toshiba-cho 1 Toshiba Corporation (DB 69-054-3517) research and development center \*

(72)【発明者】

(72) [Inventor]

【氏名】

[Name]

鈴木 真樹

Suzuki Maki

【住所又は居所】

[Address]

神奈川県川崎市幸区小向東芝町1番地 株式会社東芝研究開発センター内

Kanagawa Prefecture Kawasaki City \*\*Komukai Toshiba-cho 1 Toshiba Corporation (DB 69-054-3517) research and development center \*

**Agents**

(74)【代理人】

(74) [Attorney(s) Representing All Applicants]

【弁理士】

[Patent Attorney]

【氏名又は名称】

[Name]

鈴江 武彦

Suzue Takehiko

**Abstract**

(57)【要約】

(57) [Abstract ]

【課題】

[Problems to be Solved by the Invention ]

共通バスの使用率の向上を可能とする計算機装置を提供すること。

Offer computer device which makes improvement of usage of common bus possible.

## 【解決手段】

複数のユニット 6 と、複数のユニットを接続する共通バス VBUC と、プログラムを格納する格納手段 22 と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる割当手段 24 とを備えたことを特徴とする。

好ましくは、プログラムを作成し、前記格納手段に書き込む手段 20 をさらに備える。

また、好ましくは、前記格納手段は複数のメモリバンクからなり、あるバンクへの中央制御装置からの書き込みと、他のバンクからの前記割当手段による読み出しとが並列的に行われる。

また、好ましくは、前記割当手段は、前記プログラムに従って、各ユニットにアドレス及び動作モードを転送する。

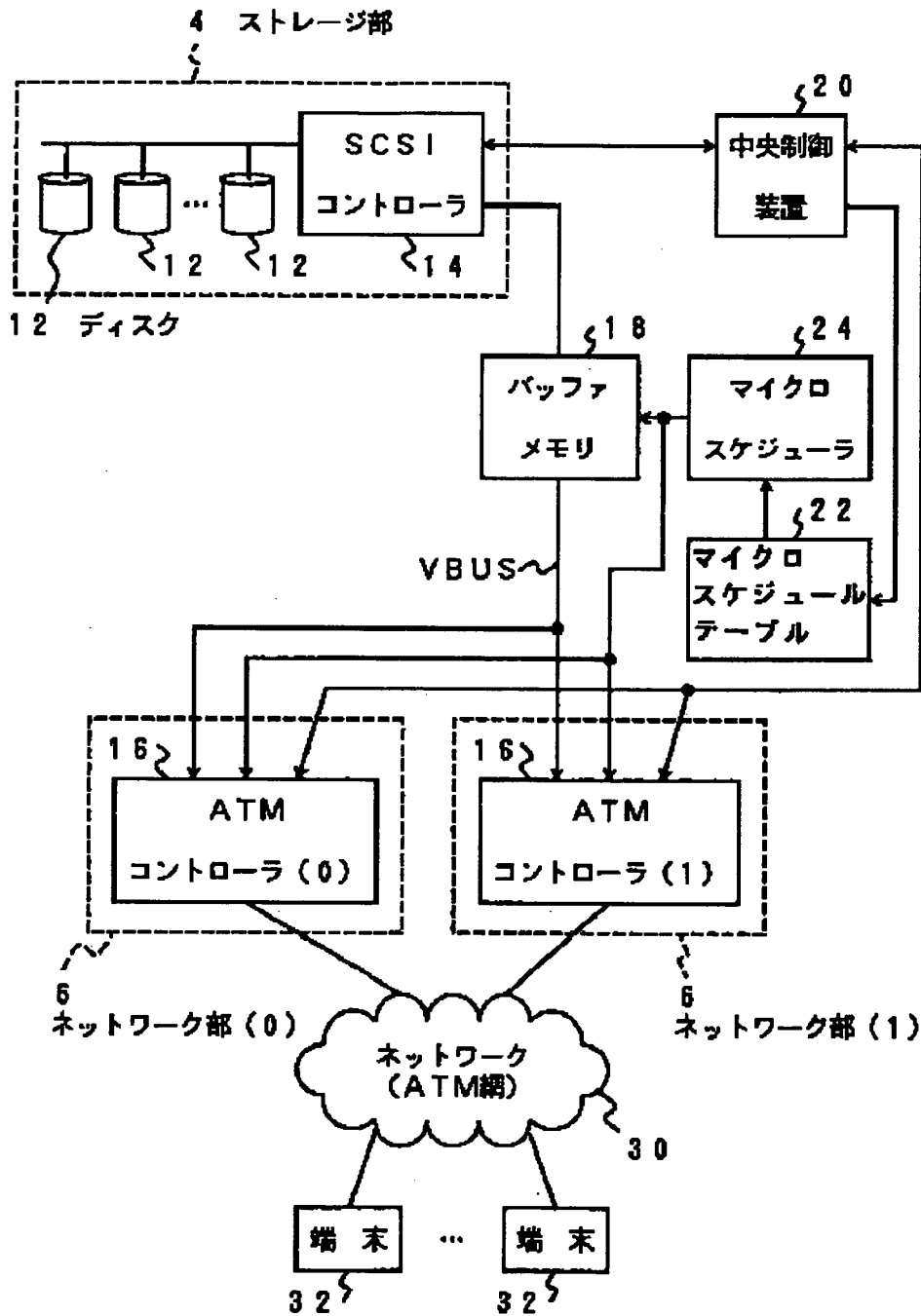
## [Means to Solve the Problems]

Following to storage means 22 and this program which house unit 6 of the plural and common bus VBUC and program which connect unit of the plural, it designates that it has allotment means 24 which allots the right each unit to use common bus, to deterministic as feature.

preferably, program is drawn up, means 20 which is written to the aforementioned storage means furthermore has.

In addition, preferably, aforementioned storage means consists of memory bank of plural, with writing from central control system to a certain bank and the aforementioned allotment means from other bank reading is done in arrayed.

In addition, preferably, aforementioned allotment means, following to aforementioned program, transfers address and operating mode in each unit.



## Claims

## 【特許請求の範囲】

## 【請求項 1】

複数のユニットと、複数のユニットを接続する共通バスと、プログラムを格納する格納手段と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる割当手段とを備えたことを特徴とする計算機装置。

## 【請求項 2】

前記プログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする請求項 1 に記載の計算機装置。

## 【請求項 3】

前記格納手段は複数のメモリバンクからなり、

あるバンクへの中央制御装置からの書き込みと、他のバンクからの前記割当手段による読み出しとが並列的に行われることを特徴とする請求項 2 に記載の計算機装置。

## 【請求項 4】

前記割当手段は、前記プログラムに従って、各ユニットにアドレス及び動作モードを転送するものであることを特徴とする請求項 1 に記載の計算機装置。

## 【請求項 5】

前記格納手段に格納されるプログラムは、読み出し先アドレス、書き込み先アドレス及び繰り返し回数を含むエントリからなり、

前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送して該アドレスをインクリメントする動作を、前記繰り返し回数に従って繰り返すものであることを特徴とする請求項 1 に記載の計算機装置。

## 【請求項 6】

前記割当手段は、書き込み先ユニットに書き込み先アドレスを含む指示を転送した場合に、この指示に対応するデータが共通バス上に存在しなかったならば、

該アドレスのインクリメントを行わず、書き込み先ユニットに対し書き込みを行わないよう指示することを特徴とする請求項 5 に記載の計算機

## [Claim (s)]

## [Claim 1]

Following to storage means and this program which house unit of the plural and common bus and program which connect unit of the plural , computer device . which designates that it has allotment means which allots right each unit to use common bus , to deterministic asfeature

## [Claim 2]

computer device . which is stated in Claim 1 which designates that furthermore it has central control system which possesses means which draws upaforementioned program , writes to aforementioned storage means asfeature

## [Claim 3]

Aforementioned storage means consists of memory bank of plural ,

computer device . which is stated in Claim 2 which designates that with the writing from central control system to a certain bank and aforementionedallotment means from other bank reading is done in arrayed asfeature

## [Claim 4]

As for aforementioned allotment means , following to theaforementioned program , computer device . which it states in Claim 1 whichdesignates that it is something which transfers address and the operating mode in each unit as feature

## [Claim 5]

program which is housed in aforementioned storage means consists of the entry which includes reading destination address , writing destination address and number of repetitions ,

As for aforementioned allotment means , transferring display toeach unit on basis of address which is included in this entry ,following operation which increment it does said address , to theaforementioned number of repetitions , computer device . which it states in Claim 1 whichdesignates that it is something which it repeats as feature

## [Claim 6]

If as for aforementioned allotment means , when display whichincludes writing destination address in unit ahead writing was transferred, the data which corresponds to this display did not exist is on common bus ,

Way does not do increment of said address , it does not do writing vis-a-vis unit ahead writing , display computer device . which isstated in Claim 5 which designates thing

装置。

【請求項 7】

前記格納手段は複数のメモリバンクからなり、

各メモリバンクにはそれぞれ対応するターゲットユニットへの指示の基となるプログラムが格納されるものであることを特徴とする請求項 1 に記載の計算機装置。

【請求項 8】

前記割当手段は前記ターゲットユニットのそれぞれが前記共通バスをインターリーブして使用するよう割り当てることを特徴とする請求項 7 に記載の計算機装置。

【請求項 9】

ストレージから読み出したデータを一時記憶するバッファメモリユニットと、

バッファメモリのデータを各ユーザを宛先として通信路に送り出す通信制御ユニットと、

これらユニットを接続する共通バスと、

プログラムを格納する手段と、

このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる手段とを備えたことを特徴とする連続データサーバ装置。

【請求項 10】

各ユーザ宛に送り出されるべきデータが一定の間隔でバッファメモリユニットから通信制御ユニットに取り込まれるように、共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする請求項 9 に記載の連続データサーバ装置。

【請求項 11】

通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、

前記スロットを複数に等分に分割した各ミニスロット内の一定の位置に、各ユーザ宛に送り出されるべきデータを通信制御ユニットに取り込むための共通バスの使用が割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする請求項 9 に記載の連続データサーバ装置。

which is done as feature

[Claim 7 ]

Aforementioned storage means consists of memory bank of plural ,

In each memory bank computer device . which is stated in Claim 1 which designates that it is something where program which becomes basis of display to target unit which corresponds respectively is housed as feature

[Claim 8 ]

As for aforementioned allotment means each one of the aforementioned target unit interleave doing aforementioned common bus , in order to use, computer device . which is stated in Claim 7 which designates that it allots as feature

[Claim 9 ]

From storage unit reading it is buffer memory unit which remembers data at one time and,

communication control unit which sends out data of buffer memory to communications line with each user as addressee and,

common bus which connects these unit and,

means. which houses program

Following to this program , continual data server device . which designates that it has means which allots right each unit to use common bus , to deterministic as feature

[Claim 10 ]

In order for data which it should you send out to each user address with fixed spacing from buffer memory unit to be taken in to the communication control unit , continual data server device . which is stated in Claim 9 which designates that furthermore it has central control system which possesses means which draws up program which allots use of common bus , writes to aforementioned storage means as feature

[Claim 11 ]

It is through communication control unit from buffer memory timing of feed of the data to communications line with slot unit means. which display is done

Aforementioned slot in fixed position inside each [minisurotto ] which is divided into equal parts in plural , in order to be able to allot the use of common bus in order to take in data which it should you send out to each user address to communication control unit , program is drawn up, Continual data server device . which is stated in Claim 9 which designates that furthermore it has central control system which

タサーバ装置。

【請求項 12】

前記中央制御装置が書き込むプログラムは、前記ミニスロット単位で作成され、

前記割当手段は、スロットの分割数分のプログラムを繰り返して用いるものであることを特徴とする請求項 11 に記載の連続データサーバ装置。

【請求項 13】

通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、

各ユーザ宛に送り出されるべきデータのビットレートに応じた数の、前記スロットを複数に等分に分割したマイクロスロットが、該データを通信制御ユニットに取り込むための共通バスの使用期間として割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする請求項 9 に記載の連続データサーバ装置。

【請求項 14】

前記中央制御装置が書き込むプログラムは、バッファメモリユニット及び通信制御ユニット内のアドレス、前記マイクロスロットの数に対応する繰り返し回数を含むエントリからなり、

前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送する動作を、前記繰り返し回数に従って繰り返すことにより、前記ビットレートに応じた使用期間を割り当てるものであることを特徴とする請求項 13 に記載の連続データサーバ装置。

【請求項 15】

ストレージから読み出されたデータをバッファメモリユニットに取り込む際の共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする請求項 9 に記載の連続データサーバ装置。

Specification

【発明の詳細な説明】

【0001】

possesses means which is written to aforementioned storage means as feature

[Claim 12 ]

program which aforementioned central control system writes is drawn up with the aforementioned [minisurotto ] unit ,

As for aforementioned allotment means , continual data server device . which is stated in Claim 11 which designates that it is something which uses number of divisions amount this program of slot over again as feature

[Claim 13 ]

It is through communication control unit from buffer memory timing of feed of the data to communications line with slot unit means. which display is done

Aforementioned slot of a quantity which responds to bit rate of the data which it should you send out to each user address Micros lot which is divided into equal parts in plural , in order to be allotted, as use period of common bus in order to take in said data to communication control unit program is drawn up, Continual data server device . which is stated in Claim 9 which designates that furthermore it has central control system which possesses means which is written to aforementioned storage means as feature

[Claim 14 ]

program which aforementioned central control system writes consists of entry which includes number of repetitions which corresponds to quantity of address , aforementioned Micros lot inside buffer memory unit and communication control unit ,

As for aforementioned allotment means , following operation which transfers display to each unit on basis of address which is included in this entry , to aforementioned number of repetitions , the continual data server device . which it states in Claim 13 which designates that it is something which allots use period which responds to the aforementioned bit rate by repeating, as feature

[Claim 15 ]

Case where data which reads out from storage unit is taken in to buffer memory unit continual data server device . which is stated in Claim 9 which designates that furthermore it has central control system which possesses means which draws up program which allots use of common bus , writes to aforementioned storage means as feature

[Description of the Invention ]

【0001】



## 【発明の属する技術分野】

本発明は、共通バスに接続された複数のユニットを持つ計算機装置および、少なくとも一部のユニット間のデータ転送に共通バスを利用する連続データサーバ装置に関する。

【0002】

## 【従来の技術】

デジタル情報を扱う計算機システムで複数のユニットを接続する安価な方法として共通バスが広く採用されている。

共通バスを使用したシステムを構築する上での問題点は共通バスのバンド幅ネックが発生し易いことである。

共通バスのバンド幅を増やすためには信号線の数を増やすか、共通バスの動作周波数を上げることが考えられるがいずれの場合にもコストの増大を招くことになる。

【0003】

コストの増大を避けながら共通バスのバンド幅ネックを防ぐ効果的な方法としては共通バスの使用率の向上が考えられる。

しかし一般に共通バスに複数のユニットを接続した場合、共通バスの使用率をめぐっての調停動作が必要であり、調停動作には一定の時間が必要なので、これが共通バスの使用率を高める上で障害となっていた。

【0004】

また、メモリを多数用いる計算機システムではメモリとして安価な DRAM が使用されることが多い。

DRAM は高速ページモードなどで使用するとバンド幅は大きくとれるが、レーテンシも大きいという問題がある。

従って、DRAM と共通バスの間でデータ転送が行われる場合、調停動作にレーテンシが加わり、さらに共通バスの使用率を低下させる。

【0005】

ところで、共通バスを用いる計算機システムの 1 つに、連続データサーバ装置がある。

映像や音声のような連続データを扱う連続データサーバ装置は、記憶装置に記憶した連続データを読み出し、端末装置に対して時間に同期しながらリアルタイムに連続的に(一定期間内に一定量の)データを送り出す機能を持つ。

## 【Technological Field of Invention】

this invention computer device which has unit of plural which isconnected to common bus and, regards continual data server device which at leastutilizes common bus in data transfer between unit of part.

【0002】

## 【Prior Art】

common bus is adopted to be wide as inexpensive method which connects unit of plural with computer system which handles digital data .

When constructing system which uses common bus problem is that the dope width neck of common bus is easy to occur.

In order to increase dope width of common bus , you can think that the operating frequency of common bus is increased whether it increases quantity of signal line , but it means to cause increase of cost to in each case.

【0003】

While avoiding increase of cost, you can think improvement of usage of common bus as effective method which prevents dope width neck of the common bus .

But when unit of plural is connected to common bus generally, arrange operation centering on use right of common bus beingnecessary, because fixed time is necessary in arrange operation, when this raises usage of common bus , it had become damage .

【0004】

In addition, among computer system which large number uses memory there is many a thing where inexpensive DRAM is used as memory .

As for DRAM when you use with high speed page mode etc, as for dope width it comes off largely, but there is a problem that also [reetenshi ] is large.

Therefore, when data transfer is done between DRAM and common bus , the[reetenshi ] joins to arrange operation, furthermore usage of common bus decreases.

【0005】

By way, in one of computer system which uses common bus , there is a continual data server device .

Continual data server device which handles continual data like image and audio while synchronization designating continual data which storage is made storage device as time vis-a-vis reading , terminal apparatus has thefunction which sends out (Inside constant period constant amount ) data to continuous in real time

このような連続データサーバ装置は、複数の映画等のビデオデータを記憶し端末からの要求に応じて任意の映画を送り出すビデオ・オン・デマンド、ネットワークを介して映像による商品情報を提供するオンラインショッピングなどの分野で使用される。

それゆえ、連続データサーバ装置には、複数のユーザからランダムに要求が送られてくるので、同時に多数のユーザに対してそれぞれ異なる連続データの送り出しを間断なく行なう能力が要求されることになる。

[0006]

以下、従来の連続データサーバ装置について詳しく説明する。

従来の連続データサーバ装置の 1 つに、図 19 に示すような構成を持つものがある。

このような連続データサーバ装置において、ユーザあるいはアプリケーションプログラムによって発せられた連続データへのアクセス要求が、プロセス間通信やネットワークを経由した通信等によって送られてくる。

このアクセス要求は、ネットワーク部 706 から共通バス 721 を介して中央制御装置 720 に伝えられ、受理される。

中央制御装置 720 は、要求された連続データの読み出しをストレージ部 704 に伝える。

ストレージ部 704 のデータ記憶制御装置 714 は、指示された連続データをデータ記憶装置 712 から読み出して、バッファメモリ 718 に書き込む。

中央制御装置 720 は、バッファメモリ 718 上のデータの送り出しをネットワーク部 706 に指示する。

ネットワーク部 706 は、アクセス要求にて指定された転送先に対し、連続データを送り出す。

これら動作は、通常、スロットと呼ばれる一定の時間間隔を単位として行われる。

連続データを記憶するデータ記憶装置には、ディスク装置を用いる場合が多いが、光ディスクや光磁気ディスク装置等を用いる場合もある。

ディスク装置以外にも、RAM や EEPROM 等の半導体記憶装置を用いることもある。

continuous in real time .

Continual data server device a this way storage does motion picture or other video data of plural and through video \*on \*demand , network which sends out motion picture of option in compliance with request from terminal is used with online shopping or other field which offers product data with image .

Consequently, because in continual data server device , from user of the plural request is sent to random , simultaneously without breakvis-a-vis multiple user does feed of respective different continual data it means that capacity which is required.

[0006]

You explain in detail below, concerning conventional continual data server device .

In one of conventional continual data server device , there are some which have kind of configuration which is shown in Figure 19 .

access demand for continual data which was given out with the user or application program in continual data server device a this way, is sent with communication between process and communication etc which goes by way of the network .

this access request, through common bus 721 from network section 706, is conveyed by central control system 720, acceptance is done.

central control system 720 conveys reading of continual data which is required to storage unit section 704.

As for data storage control device 714 of storage unit section 704, continual data which the display is done reading \*, is written to buffer memory 718 from data storage device 712.

feed of data on buffer memory 718 display it designates central control system 720, as network section 706.

network section 706 sends out continual data vis-a-vis forwarding destination which is appointed with access request.

These operations are done fixed time interval which usually, is called the slot as unit .

When disk drive is used is many in data storage device which remembers the continual data , but when optical disk and magneto-optical disk device etc are used, it is.

It is in addition to disk drive , also times when RAM and the EEPROM or other semiconductor storage device are used.

[0007]

また、図 20 に示すように複数系列のストレージ部 704 を備え、1 つの連続データを分割して複数系列に渡って分散格納するストライピング技法を適用したものがあ

これは、データ転送能力(総バンド幅)を大きくし、同一の連続データへより多数のユーザが同時にアクセスすることを可能にすることを目的としている。

[0008]

この種の従来の連続データサーバ装置では、ストレージ部 704 やネットワーク部 706 が共通バス 721 の使用要求を出し、図示しない調停装置が調停を行い、使用権を獲得したストレージ部 704 やネットワーク部 706 が共通バス 721 を使用してバッファメモリ 718 との間のデータ転送を行う。

従って、調停動作に要する時間が共通バス 721 の使用率を低下させ、同時にアクセスできるユーザ数が低下されてしま

また、連続データサーバ装置ではユーザ端末側での再生を中断させないために、各ユーザ端末に一定の時間間隔で一定量のデータを間断なく転送することを常に保証する必要があるので、この保証のために、バスの使用率を余裕をとって低く抑えておく。

従って、上記保証をしつつ、共通バスの使用率を向上させる新たな技術が望まれている。

[0009]

【発明が解決しようとする課題】

従来、共通バスに接続された複数のユニットを持つ計算機装置では、共通バスの使用権をめぐっての調停動作に一定の時間が浪費され、これが共通バスの使用率を高める上で障害となっていた。

また、DRAM と共通バスの間でデータ転送が行われる場合、調停動作にレーテンシが加わり、さらに共通バスの使用率を低下させる問題があった。

[0010]

また、従来、装置内部のユニット間データ転送に共通バスを利用する連続データサーバ装置では、あるユーザ端末について一定の時間間隔で一定量のデータを間断なく転送することを常に保証しつつ、共通バスの使用率を向上させることは困難であった。

[0007]

In addition, as shown in Figure 20, there are some which apply [sutoraipingu] technique where it has storage unit section 704 of plural array, divides the continual data of one and it disperses it houses over the plural array.

This enlarges data transfer capacity (Entire dope width), compared to multiple user simultaneously has designated that it makes that access it does possible as objective to same continual data.

[0008]

With conventional continual data server device of this kind, storage unit section 704 and network section 706 makes use demand for common bus 721, storage unit section 704 where unshown arrange device arrangements, acquires useright and network section 706 using common bus 721, data transfer between the buffer memory 718 is done.

Therefore, time when it requires in arrange operation usage of common bus 721 decreasing, simultaneously number of users which access it is possible decreases.

In addition, because with continual data server device because regeneration on user terminal side is not discontinued, it is necessary normally to guarantee that in each user terminal data of constant amount is transferred with the fixed time interval without break, for this guarantee, taking Yutaka excessively, you hold down usage of bus low.

Therefore, while doing above-mentioned guarantee, new technology which improves is desired usage of common bus.

[0009]

【Problems to be Solved by the Invention】

Until recently, with computer device which has unit of plural which is connected to common bus, fixed time was wasted by arrange operation, centering on use right of common bus when this raises usage of common bus, had become damage.

In addition, when data transfer is done between DRAM and common bus, [reetenshi] joined to arrange operation, furthermore usage of common bus was a problem which decreases.

[0010]

In addition, until recently, with continual data server device which in the data transfer between unit of device interior utilizes common bus, data of constant amount while normally guaranteeing that is transferred with the fixed time interval without break concerning a certain user terminal, usage of the common bus as for improving it was difficult.

[0011]

本発明は、上記事情に鑑みてなされたものであり、共通バスの使用率の向上を可能とする計算機装置を提供すること目的とする。

また、本発明は、各ユーザ端末に一定の時間間隔で一定量のデータを間断なく転送することを常に保証しつつ、装置内部でのデータ転送に用いる共通バスの使用率の向上を可能とする連続データサーバ装置を提供すること目的とする。

[0012]

[課題を解決するための手段]

本発明(請求項 1)に係る計算機装置は、複数のユニット(下記マイクロスケジューラの割り当てに従って共通バスを使用する)と、複数のユニットを接続する共通バスと、プログラムを格納する格納手段(マイクロスケジューラテーブル)と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる割当手段(マイクロスケジューラ)とを備えたことを特徴とする。

[0013]

各ユニットからのバス使用要求に依存してバス使用権の調停を行うのではなく、予め定めたスケジューリングによりバス使用権を割り当てるため、各ユニットのバス使用時期を確定的に保証することができる。

さらに、これを利用して(メモリへの指示を先行発行することにより)、メモリのレーテンシを隠蔽することができる。

従って、バスの効率的な使用(使用率の向上)が実現できる。

[0014]

本発明(請求項 2)は、請求項 1 において、前記プログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする。

本発明(請求項 3)は、請求項 2 において、前記格納手段は複数のメモリバンクからなり、あるバンクへの中央制御装置からの書き込みと、他のバンクからの前記割当手段による読み出しとが並行的に行われることを特徴とする。

[0015]

本発明(請求項 4)は、請求項 1 において、前記割当手段は、前記プログラムに従って、各ユニ

[0011]

As for this invention, considering to above-mentioned situation, beingsomething which it is possible, it makes thing objective which offers computer device which makes improvement of usage of common bus possible.

In addition, this invention while normally guaranteeing that in each user terminal data of constant amount is transferred with fixed time interval withoutbreak, makes thing objective which offers continual data server device whichmakes improvement of usage of common bus which it uses for the data transfer with device interior possible.

[0012]

[Means to Solve the Problems ]

computer device relating to this invention (Claim 1 ), unit of plural (Following to allotment of below-mentioned [maikurosukejuura ], you use the common bus ) with,storage means which houses common bus and program which connect unit of plural ( [maikurosukejuuruteeburu ] ) with, following to this program , designates that it hasallotment means ( [maikurosukejuura ] ) which allots right each unit to use the common bus , to deterministic as feature.

[0013]

Depending on bus use request from each unit , it is not toarrangement bus use right , in order to allot bus use right with scheduling which it decides beforehand, you can guarantee bus use time ofeach unit decide.

Furthermore, (In preceding issuing display to memory to depend ), [reetenshi ] of memory hiding is possible making useof this.

Therefore, it can actualize efficient use (Improvement of usage ) of bus .

[0014]

this invention (Claim 2 ) draws up aforementioned program in Claim 1 ,designates that furthermore it has central control system which possesses means which is written to aforementioned storage means as feature.

As for this invention (Claim 3 ), as for aforementioned storage means it consists of memory bank of plural in Claim 2 , it designates that with writing from central control system to a certain bank and aforementioned allotment means from other bank reading is done in arrayed as feature.

[0015]

As for this invention (Claim 4 ), in Claim 1 , as for aforementioned allotment means , following to

ットにアドレス(読み出し先アドレス/書き込み先アドレス)及び動作モード(読み出し命令/書き込み命令)を転送するものであることを特徴とする請求項 1 に記載の計算機装置。

[0016]

本発明(請求項 5)は、請求項 1 において、前記格納手段に格納されるプログラムは、読み出し先アドレス、書き込み先アドレス及び繰り返し回数を含むエントリからなり、前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送して該アドレスをインクリメントする動作を、前記繰り返し回数に従って繰り返すものであることを特徴とする、本発明(請求項 6)は、請求項 5 において、前記割当手段は、書き込み先ユニットに書き込み先アドレスを含む指示を転送した場合に、この指示に対応するデータが共通バス上に存在しなかったならば、該アドレスのインクリメントを行わず、書き込み先ユニットに対し書き込みを行わないよう指示することを特徴とする。

[0017]

本発明(請求項 7)は、請求項 1 において、前記格納手段は複数のメモリバンクからなり、各メモリバンクにはそれぞれ対応するターゲットユニットへの指示の基となるプログラムが格納されるものであることを特徴とする。

[0018]

本発明(請求項 8)は、請求項 7 において、前記割当手段は前記ターゲットユニットのそれぞれが前記共通バスをインタリーブして使用するよう割り当てることを特徴とする。

[0019]

本発明(請求項 9)に係る連続データサーバは、ストレージから読み出したデータを一時記憶するバッファメモリユニットと、バッファメモリのデータを各ユーザを宛先として通信路に送り出す通信制御ユニット(これらユニットは下記のマイクロスケジューラの割り当てに従って共通バスを使用すると、これらユニットを接続する共通バスと、プログラムを格納する手段と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる手段とを備えたことを特徴とする。

[0020]

請求項 1 の発明について説明した作用効果に加え、一定の間隔で通信制御ユニットに連続データが送り込まれ、ひいては、一定の間隔で通

forementioned program , in each unit address (reading destination address /writing destination address ) and computer device . which is stated in Claim 1 which designates that it is something which transfers operating mode (reading command /writing command ) as feature

[0016]

Transferring display to each unit on basis of address which as for this invention (Claim 5 ), as for program which is housed in the aforementioned storage means in Claim 1 , consists of entry which includes reading destination address , writing destination address and number of repetitions , as for aforementioned allotment means , is included in this entry , operation which increment does said address , Following to aforementioned number of repetitions , it designates that it is something which it repeats as feature, if as for this invention (Claim 6 ), as for aforementioned allotment means , when it transferred display which includes writing destination address in unit ahead writing , data which corresponds to this display did not exist is on common bus in Claim 5 , increment of said address action, In order not to do writing , vis-a-vis unit ahead writing it designates that display it does as feature.

[0017]

As for this invention (Claim 7 ), as for aforementioned storage means it consists of memory bank of plural in Claim 1 , it designates that it is something where program which becomes basis of display to the target unit which corresponds respectively is housed as feature in each memory bank .

[0018]

As for this invention (Claim 8 ), as for aforementioned allotment means each one of aforementioned target unit interleave doing aforementioned common bus , in order to use, it designates that it allots as feature in the Claim 7.

[0019]

As for continual data server relating to this invention (Claim 9 ), reading it is communication control unit which sends out data of buffer memory unit and buffer memory which remember data at one time to communications line with each user as the addressee (These unit following to allotment of below-mentioned [maikurosukejuura ], use common bus ) with, following to means. this program which houses common bus and the program which connect these unit from storage unit , right each unit to use common bus , It designates that it has means which is allotted to deterministic as feature.

[0020]

Continual data is sent by communication control unit with fixed spacing in addition to acting effect which is explained concerning invention of Claim 1 , you can guarantee that

信路に連続データが送り出されることを保証することができる。

従って、通信制御ユニットが通信路に連続データを送り出すまで該データを一時記憶するパケットメモリの容量を少なくでき、また、通信路の先に存在して連続データを再生するユーザ端末において受信したパケットを保持するバッファの容量を少なくできる。

[0021]

本発明(請求項 10)は、請求項 9 において、各ユーザ宛に送り出されるべきデータが一定の間隔でバッファメモリユニットから通信制御ユニットに取り込まれるように、共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする。

[0022]

本発明(請求項 11)は、請求項 9 において、通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、前記スロットを複数に等分に分割した各ミニスロット内の一定の位置に、各ユーザ宛に送り出されるべきデータを通信制御ユニットに取り込むための共通バスの使用が割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする。

[0023]

これにより、スロットよりも細かい単位の一定の間隔で、通信制御ユニットに連続データが送り込まれることを保証することができる。

従って、パケットメモリやユーザ端末のバッファの容量をさらに少なくできる。

[0024]

本発明(請求項 12)は、請求項 11 において、前記中央制御装置が書き込むプログラムは、前記ミニスロット単位で作成され、前記割り当て手段は、スロットの分割数分のこのプログラムを繰り返して用いるものであることを特徴とする。

[0025]

これにより、前記格納手段の容量を小さくできる。

本発明(請求項 13)は、請求項 9 において、通信制御ユニットを介したバッファメモリから通信路

continual data is sent out to communications line with consequently , fixed spacing .

Therefore, until communication control unit sends out continual data to communications line ,it can make capacity of buffer which keeps packet which itreceives capacity of packet memory which remembers said data at one timeit can make little, in addition, communications line existing first, in the user terminal which regeneration does continual data little.

[0021]

this invention (Claim 10 ) in order for data which it should you send out toeach user address in Claim 9, with fixed spacing from buffer memory unit to be taken in to communication control unit , draws up program which allots useof common bus , designates that furthermore it has central control system whichpossesses means which is written to aforementioned storage means asfeature.

[0022]

communication control unit it is through this invention (Claim 11 ), in Claim 9, from buffer memory the timing of feed of data to communications line with slot unit means. aforementioned slot which display is done in fixed position insideeach [minisurotto ] which is divided into equal parts in plural , In order to be able to allot use of common bus in order to take in data being supposed you send out to each user address to the communication control unit , it draws up program , it designates that furthermore it has central control system which possesses means which is written to theaforementioned storage means as feature.

[0023]

Because of this, with fixed spacing of small unit , you canguarantee that continual data is sent to communication control unit in comparisonwith slot .

Therefore, capacity of buffer of packet memory and user terminal furthermore can be made little.

[0024]

As for this invention (Claim 12 ), as for program which aforementioned central control system writes in Claim 11 , it is drawn up with aforementioned [minisurotto ] unit ,aforementioned allotment means designates that it is somethingwhich uses number of divisions amount this program of slot over again as feature.

[0025]

Because of this, capacity of aforementioned storage means can be madesmall.

As for this invention (Claim 13 ), in Claim 9, it is through communication control unit from the buffer memory ,

へのデータの送り出しのタイミングをスロット単位で指示する手段と、各ユーザ宛に送り出されるべきデータのビットレートに応じた数の、前記スロットを複数に等分に分割したマイクロスロットが、該データを通信制御ユニットに取り込むための共通バスの使用期間として割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする。

【0026】

これにより、各ユーザの要求ビットレートに応じたバス使用期間を、各スロット内で確保することができる。

クレーム 11 のミニスロットと組み合わせて(ミニスロットをさらに分割したものをマイクロスロットとして)用いるとさらに良い。

【0027】

本発明(請求項 14)は、請求項 13 において、前記中央制御装置が書き込むプログラムは、バッファメモリユニット及び通信制御ユニット内のアドレス、前記マイクロスロットの数に対応する繰り返し回数を含むエントリからなり、前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送する動作を、前記繰り返し回数に従って繰り返すことにより、前記ビットレートに応じた使用期間を割り当てるものであることを特徴とする。

【0028】

これにより、前記格納手段の容量を小さくできる。

本発明(請求項 15)は、請求項 9 において、ストレージから読み出されたデータをバッファメモリユニットに取り込む際の共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする。

また、請求項 10 の中央制御装置に本手段を備えて構成することもできる。

【0029】

なお、請求項 2~8 の各発明は、請求項 9 の連続データサーバ装置にも適用可能である。

請求項 6 を請求項 9 に適用する場合、前記割当手段は、バッファメモリユニットに書き込み先アドレスを含む指示を転送した場合に、この指示に対応するデータが共通バス上に存在しなかった

forementioned slot of a quantity which to bit rate of the data which it should you send out to means. each user address which timing of feed of data to communications line with slot unit display is done responds Micros lot which is divided into equal parts in plural , In order to be allotted, as use period of common bus in order to take in said data to communication control unit it draws up program , it designates that furthermore it has central control system which possesses means which is written to aforementioned storage means as feature.

【0026】

Because of this, bus use period which responds to request bit rate of each user , can be guaranteed inside each slot .

When ( Those which furthermore divide [minisurotto ] Micros lot doing ) it uses claim 11 [minisurotto ] with combining furthermore it is good.

【0027】

Operation which transfers display to each unit on basis of address which as for this invention (Claim 14 ) , as for program which the aforementioned central control system writes in Claim 13 , consists of entry which includes number of repetitions which corresponds to quantity of address , aforementioned Micros lot inside buffer memory unit and communication control unit as for the aforementioned allotment means , is included in this entry , Following to aforementioned number of repetitions , it designates that it is something which allots use period which responds to aforementioned bit rate by repeating, as feature.

【0028】

Because of this, capacity of aforementioned storage means can be made small.

this invention (Claim 15 ) case where data which reads out from storage unit in Claim 9 , is taken in to buffer memory unit draws up program which allots use of common bus , designates that furthermore it has central control system which possesses means which is written to aforementioned storage means as feature.

In addition, providing this means for central control system of Claim 10 , the configuration it is possible also to do.

【0029】

Furthermore, each invention of Claim 2 ~8 is applicable even in the continual data server device of Claim 9.

When Claim 6 is applied to Claim 9 , if as for aforementioned allotment means , when display which includes writing destination address in buffer memory unit was transferred, data which corresponds to this display did

ならば、該アドレスのインクリメントを行わず、バッファメモリユニットに対し書き込みを行わないよう指示するようにする。

【0030】

請求項7を請求項9に適用する場合、前記格納手段は複数の通信制御ユニット対応に設けられたメモリバンクからなり、各メモリバンクにはそれぞれ対応する通信制御ユニットへの指示の基となるプログラムが格納されるようにする。

【0031】

請求項8を請求項9に適用する場合、前記割当手段は前記通信制御ユニットへのそれぞれに対する指示をインタリーブして転送するようにする。

また、請求項11~14の各発明は、請求項15の連続データサーバ装置にも適用可能である。

これらの場合、請求項15の中央制御装置は、ストレージからバッファメモリユニットへのデータの読み出しのタイミングをスロット単位で指示する手段をさらに含み、前記共通バスの使用の割り当ては、前記スロットを複数に等分に分割したマイクロスロット単位で行っても良い。

【0032】

なお、前記格納手段はメモリを用いて構成しても良い。

また、前記格納手段に格納するプログラムとして何もしないNOP命令を格納できるようにしても良い。

【0033】

また、各ユニットにアドレス及び動作モードを転送するために、データを転送する転送路(共通バス)と同じ転送路を使用しても良いし、該データを転送する転送路とは別に設けた転送路を使用しても良い。

【0034】

【発明の実施の形態】

以下、図面を参照しながら発明の実施の形態を説明する。

本実施形態では、本発明に係るマイクロスケジューラによる決定論的な割り当てに従って共通バスを使用する複数のユニットを有する計算機装置として、複数の連続データのアクセス要求に同時に応答して映像や音声等の連続データのサービスを行う連続データサーバ装置を取り

notexist is on common bus , it does not do increment of said address , in ordernot to do writing , vis-a-vis buffer memory unit display it does, itrequires.

[0030]

When Claim 7 is applied to Claim 9, as for aforementioned storage means it consists of memory bank which is provided in communication control unit correspondence of plural , program which becomes basis of the display to communication control unit which corresponds respectively is housed requiresin each memory bank .

[0031]

When Claim 8 is applied to Claim 9, interleave doing display forrespectively to aforementioned communication control unit it transfers theaforementioned allotment means it requires.

In addition, each invention of Claim 11 ~14 is applicable even in thecontinual data server device of Claim 15 .

In these cases, central control system of Claim 15 , from storage unit timing of reading of data to buffer memory unit furthermore including means which display is done with slot unit , aforementioned slot with Micros lot unit which is divided into equal parts in plural is goodallotting use of aforementioned common bus .

[0032]

Furthermore, aforementioned storage means configuration is good doing makinguse of memory :

In addition, NOPcommand which nothing is done as program which ishoused in aforementioned storage.means can be housed is good requiring.

[0033]

In addition, transfer road which transfers data in order tottransfer address and operating mode in each unit , (common bus ) with it is goodusing same transfer road and, it is good using transfer roadwhich is provided separately from transfer road which transfers said data .

[0034]

[Embodiment of the Invention ]

While below, referring to drawing , you explain Embodiment of Invention .

With this embodiment , deterministic following to allotment with [maikuroskujeuura ] whichrelates to this invention , responding simultaneously to access demandfor continual data of plural as computer device which possesses the unit of plural which uses common bus , it picks up continual data server device which does service of image and audio or other



上げる。

[0035]

第 1 の実施形態および第 2 の実施形態では本発明を連続データサーバのネットワーク部に適用した場合について、第 3 の実施形態では本発明を連続データサーバ装置のネットワーク部およびストレージ部に適用した場合について説明する。

[0036]

(実施形態 1)最初に、実施形態 1 について説明する。

図 1 に本実施形態に係る連続データサーバ装置の構成を示す。

[0037]

本実施形態の連続データサーバ装置は、連続データを格納している所定台数のデータ記憶装置 12 と該データ記憶装置 12 からデータを読み出すデータ記憶制御装置 14 とからなるストレージ部 4、該ストレージ部 4 から読み出したデータを一時記憶するバッファメモリ 18、該バッファメモリ 18 のデータを各ユーザ端末 32 を宛先としてネットワーク 30 に送り出す複数(ここでは 2 台とする)のネットワーク部 6、ストレージ部 4 とバッファメモリ 18 とネットワーク部 6 を接続する共通バス VBUS、端末 32 からの要求に基づいてデータ記憶装置 12 からデータを読み出してネットワーク 30 に送り出すまでのスケジューリング、ネットワークの設定をはじめとして、システム全体の制御を行なう中央制御装置 20、該中央制御装置 20 により確定されたマイクロスケジューリングのプログラムを格納するマイクロスケジューリングテーブル 22、該プログラムに従ってネットワーク部 6 に共通バス VBUS を使用する権利を確定的に割り当てるマイクロスケジューラ 24 を備えている。

[0038]

ストレージ部 4、ネットワーク部 6、バッファメモリ 18 の役割は夫々図 19 や図 20 の対応するユニットと基本的には同様であるが、本実施形態では、中央制御装置 20 はストレージ部 4 とネットワーク部 6 にスロット(一定の時間間隔)単位の動作指令を出すとともに、マイクロスケジューリングテーブル 22 にマイクロスケジューリングのプログラムを書き込み、マイクロスケジューラ 22 はこのプログラムに従いストレージ部 4 およびネットワーク部 6 による共通バス VBUS の使用を確定的に制御していく。

すなわち、各ユニットからのバス使用要求に依存してバス使用権の調停を行うのではなく、予

continual data .

[0035]

When with first embodiment and second embodiment this invention is applied to network section of continual data server , being attached, when with embodiment of third this invention it applies to network section and storage unit section of continual data server device , being attached, you explain.

[0036]

(embodiment 1 ) First, you explain concerning embodiment 1 .

configuration of continual data server device which relates to this embodiment in the Figure 1 is shown.

[0037]

As for continual data server device of this embodiment , storage unit section 4 which consists of data storage controller 14 which read-out does data from data storage device 12 and said data storage device 12 of specified number of devices which houses continual data , from said storage unit section 4 reading it is network section 6 of plural (Here 2 it does ) which sends out data of buffer memory 18 , said buffer memory 18 which remembers data at one time to network 30 with each user terminal 32 as addressee , Until data reading \* is sent out to network 30 from data storage device 12 storage unit section on basis of request from common bus VBUS , terminal 32 which connects 4 and buffer memory 18 and network section 6 , program of [maikurosukeyuuru ] which is decided by central control system 20 , said central control system 20 which controls system entirety with the setting of scheduling , network as beginning , is housed [maikurosukeyuuru ] 22 , Following to said program , decide it allots right to use common bus VBUS for network section 6 [maikurosukeyuuru ] it has 24 .

[0038]

storage unit section 4 , network section 6 , role of buffer memory 18 is similar to unit and basic to which respectively Figure 19 and Figure 20 correspond , but with this embodiment , as for central control system 20 as in storage unit section operation command of slot (Fixed time interval ) unit is put out 4 and network section 6 , [maikurosukeyuuru ] In 22 program of [maikurosukeyuuru ] writing , [maikurosukeyuuru ] 22 controls use of the common bus VBUS decide in storage unit section 4 and network section 6 in accordance with this program .

Depending on bus use request from namely , each unit , it is not to arrangement bus use right , bus use time of each unit

め定めたマイクロスケジュールによりバス使用権を割り当てることにより、各ユニットのバス使用時期を確定的に保証するようにしている。

これによって、一定の間隔でネットワーク部 6 に一定量の連続データが送り込まれ、一定の間隔で通信路に一定量の連続データが送り出されることを保証しつつ、共通バスの使用率の向上を図ることができ、同時サービス可能なユーザ数を向上させることができる。

また、ネットワーク部 6 が通信路に連続データを送り出すまで該データを一時記憶するパケットメモリ 18 の容量を少なくでき、また、通信路の先に存在して連続データを再生するユーザ端末において受信したパケットを保持するバッファの容量を少なくできる利点もある。

[0039]

以下、本実施形態をさらに詳しく説明していく。

まず、中央制御装置 20、ストレージ部 4、ネットワーク部 6 の基本的な構成について説明する。

[0040]

中央制御装置 20 は、例えば電子計算機と同じように CPU とメモリ装置から構成し、システム全体に対する制御を記述したソフトウェアを CPU で実行することにより、その機能を得ることができる。

中央制御装置 20 は、システム全体を制御するために、システム内に記憶している各連続データの仕様、各連続データのディスク装置 12 への配置状態、各ネットワーク部 6 が接続できる通信路など、システム内の情報をすべて管理している、あるいは知ることができる。

連続データの仕様としては、連続データ名あるいは ID コードなどの各連続データを特定するための情報の他に、例えば各連続データの全データ長や、連続データが複数のブロックからなる場合の全ブロック数などが考えられる。

[0041]

データ記憶装置 12 は、映像や音声等の連続データを記憶するためのものであり、磁気ディスク装置、光ディスク装置、光磁気ディスク装置等のディスク装置を用いることができる。

また、ディスク装置以外にも、RAM や EEPROM 等の半導体記憶装置など種々のものを用いることができる。

データ記憶装置 12 に記憶する連続データは、

isguaranteed decide by allotting bus use right with [maikurosukeyuuru ] which is decidedbeforehand, it has required.

Now, continual data of constant amount is sent by network section 6 withfixed spacing , while guaranteeing that continual data of the constant amount is sent out to communications line with fixed spacing , can assure theimprovement of usage of common bus , simultaneous service possible number of users can improve.

In addition, until network section 6 sends out continual data to communications line , there is also a benefit which can make capacity of buffer which keeps packet which it receives capacity of packet memory 18 whichremembers said data at one time in user terminal which it can make little,in addition, communications line existing first, regeneration does continual data little.

[0039]

Below, this embodiment is explained furthermore in detail.

First, you explain central control system 20, storage unit section 4, concerning basic constitution of network section 6.

[0040]

configuration it does central control system 20, from similar CPU , to for example electronic computer and memory device it can acquire function by executing software whichdescribes control for system entirety with CPU .

central control system 20, in order to control system entirety , has managed, data inside system such as communications line which can connect arrangement state , each network section 6 to disk drive 12 of specification , each continual data of eachcontinual data which storage has been made inside system entirety, or it can know.

As specification of continual data , to other than data in orderspecific to do continual data name or IDcord or other each continual data ,you can think all data length of for example each continual data and thequantity etc of entire block when continual data consists of block of plural .

[0041]

data storage device 12, with those in order to remember image and audio or other continual data , can use magnetic disk device , optical disk device , magneto-optical disk device or other disk drive .

In addition, various ones such as RAM and EEPROM or other semiconductor storage device can be used inaddition to disk drive .

Continual data which is remembered in data storage device 12

連続したビットあるいはバイトの並んだ構造を持つデータである。

連続データは、好ましくは、ブロックなど一纏まりの単位で記憶・管理される。

また、データ記憶制御装置 14 は夫々、データ記憶装置 12 に記憶された連続データを読み出し、これをバッファメモリ 18 の指示されたアドレスに書き込む。

本実施形態では、データ記憶装置 12 として SCSI I インタフェースを持つ磁気ディスク装置を用いるものとし、以下、データ記憶装置 12 をディスク 12、データ記憶制御装置 14 を SCSI コントローラ 14 と呼んで、説明を行うものとする。

なお、連続データの代表例はビデオ・データであり、この場合連続データサーバ装置は一般的にビデオサーバと呼ばれる。

[0042]

ネットワーク部 6 は、バッファメモリ 18 の指定されたアドレスから連続データを読み出し、これをネットワーク 30 の通信路に対して送り出す。

ネットワーク部 6 には、ATM 網やイーサネット、FDDI などを接続することができる。

本実施形態では、ネットワーク 30 を ATM 網とし、ネットワーク部 6 は ATM コントローラ 16 を用いて構成するものとして説明するが、他のプロトコルに基づくネットワークに接続する場合は、接続するネットワークに応じたインタフェース機能をネットワーク部 6 内に適宜設ければ良い。

[0043]

ここで、本実施形態のネットワーク部 6 の構造について説明する。

図 2 にネットワーク部 6 の内部構成の一例を示す。

図のようにネットワーク部 6 は ATM コントローラ 16 およびネットワークプロセッサ 162 から構成される。

[0044]

ネットワークプロセッサ 162 は、中央制御装置 20 の指示に基づいて、ATM コントローラ 16 のコントロール・メモリ(Control Memory)166 の設定を行なう。

コントロール・メモリ 166 には、ATM の回線の設定、パケットメモリ内のバッファの設定などの情報を設定する。

is data which has structure where bit or byte which is continued lines up.

Continual data storage \* is managed with unit of one Matome ballsuch as preferably , block .

In addition, as for data storage controller 14 continual data which is remembered in respectively , data storage device 12 is written to address which display of buffer memory 18 is done reading , this.

With this embodiment , use magnetic disk device which has SCSI interface as data storage device 12 below, reading disk 12, data storage controller 14 SCSI controller 14, data storage device 12 explain.

Furthermore, as for representative example of continual data with video \* data , in case of this continual data server device is called video server generally.

[0042]

network section 6 continual data sends out reading , this from the address where buffer memory 18 is appointed vis-a-vis communications line of network 30.

ATM network and Ethernet , FDDI etc can be connected to network section 6.

With this embodiment , it designates network 30 as ATM network , as for network section 6 it explains as those which configuration are done making use of ATM controller 16, but case you connect to network which is based on the other protocol , interface function which responds to network which is connected as needed it provides inside network section 6, it is good.

[0043]

Here, you explain concerning structure of network section 6 of the this embodiment .

one example of internal configuration of network section 6 is shown in Figure 2 .

As in figure network section 6 configuration is done from ATM controller 16 and network processor 162.

[0044]

network processor 162 control & memory of ATM controller 16 (control memory ) sets 166 on basis of display of central control system 20.

Setting or other data of buffer while setting and packet memory of the circuit of ATM is set to control & memory 166.

バッファメモリ 18 からのデータは VBUS インタフェース部(以下、VBUSIF と記す)163 を経由して ATM コントローラ 16 のパケットメモリ 164 に一旦蓄積される。

パケットメモリ 164 に送られたデータは、SAR チップ部(以下、SARCHIP と記す)165 により ATM パケットに組み立てられ物理層の信号に変換された後、物理層インタフェース部 167 を介して ATM 網 30 に送り出される。

[0045]

以下では、マイクロスケジューラ 24 に関して順次説明していく。

まず、本実施形態の概略動作は以下のようになる。

ユーザ端末 32 から連続データサーバ装置に対して連続データの再生要求が発行されたとき、中央制御装置 20 はストレージ部 4 のどのディスク装置 12 のどの部分に要求された連続データが存在するかを検索し、ディスクアクセスが他のユーザ端末からの要求と競合しないようにディスクアクセスのスケジュールを作成する。

[0046]

なお、一般に、ディスク装置 12 は連続した大きな単位をアクセスするとアクセスの効率がよいので、連続データサーバ装置ではデジタル化され圧縮されたデータを比較的大きな単位で(例えば 128KByte)バッファメモリ 18 に読み出し、これを例えば 4Mbps 程度で読み出す場合には 250msec 程度かかって再生する。

ここで、128KByte のディスクのアクセスは再生の速度 250msec より十分速く、例えば 60msec 程度で終了することに着目すれば、1 台のディスク装置で複数の連続データの要求を同時に扱えることがわかる。

[0047]

中央制御装置 20 では、スケジューラと呼ばれるソフトウェアによって複数のユーザからの要求を調停し、ディスク装置 12 でアクセスの競合がないように効率良いスケジューリングを行なって多数のユーザからの要求に対してサービスを行なう。

制御を簡単にするためにスケジューラはスロットと呼ばれる一定の時間間隔を単位として、例えばスロットの切れ目毎に次のスロットのスケジューリングを決定し、ストレージ部 4 の SCSI コントローラ 14 に対して指令を与える。

data from buffer memory 18 compilation makes once packet memory 164 of ATM controller 16 VBUSinterface (Below, VBUSIF you inscribe. ) via 163.

It unites data which is sent to packet memory 164, to ATM packet SARTip (Below, SARCHIP you inscribe. ) with 165 and stand and others \* after being converted to signal of physical layer, through physical layer interface 167, it is sent out to the ATM network 30.

[0045]

At below, sequential you explain [maikurosukeyuura ] in regard to 24.

First, outline operation of this embodiment is below, it groans.

When regeneration demand for continual data is issued from user terminal 32 vis-a-vis continual data server device , central control system 20 searches, continual data which is required to which portion of which disk drive 12 of storage unit section 4exists, in order for disk access not to compete with request fromother user terminal , draws up schedule of disk access .

[0046]

Furthermore, because generally, as for disk drive 12 when continued thelarge unit is done access , efficiency of access is good, withoutcontinual data server device digitization it is done and when data which wascompressed with relatively large unit in (for example 128KByte ) buffer memory 18 reading , thisis read out at for example 4Mbps extent 250 msec extent being required, regeneration it does.

If you pay attention to here, access of disk of 128 KByte the fully being quicker than speed 250msec of regeneration , ending at for example 60msec extent, it understands that demand for continual data of the plural can be handled simultaneously with disk drive of 1.

[0047]

With central control system 20, [sukejuura ] with request from user of plural will bearrangemented with software which is called, doing efficient scheduling where competition of access will be with disk drive 12, it does service vis-a-vis request from multiple user .

In order to make control simple [sukejuura ] decides scheduling of thefollowing slot in every cut of for example slot with fixed time interval whichis called slot as unit , gives command vis-a-vis SCSI controller 14 of storage unit section 4.

【0048】

ユーザ端末 32 からの再生要求に基づいて前述のケジューリングを行った中央制御装置 20 から指令を与えられたストレージ部 4 の SCSI コントローラ 14 は、ディスク装置 12 から要求されたデータを読み出し、読み出されたデータをバッファメモリ 18 の指定アドレスへ順次格納する。

【0049】

同様に中央制御装置 20 はスロット単位に端末 32 にデータを送出するスケジューリングを決定し、ネットワーク部 6 の ATM コントローラ 16 に対して指令を与え、バッファメモリ 18 はマイクロスケジューラ 24 の制御に従ってデータを読み出し、ネットワーク部 6 はマイクロスケジューラ 24 の制御に従って、読み出されたデータを取り込む。

【0050】

そして、ネットワーク部 6 からネットワーク 30 に各要求元のユーザ端末 32 に宛てて、一定期間ごとに一定量の連続データが送出される。

さて、マイクロスケジューラ 24 は、マイクロスケジューラテーブル 22 に格納されたマイクロスケジューラのプログラムに従って、バッファメモリ 18 とネットワーク部 6 を接続する共通バス VBUS の使用を制御する。

すなわち、マイクロスケジューラ 24 は、ターゲットユニットに対する動作命令を含む所定のフォーマット(図 4 参照)の制御命令をマイクロスケジューラテーブル 22 内のテーブルのエントリから 1 つずつ読み出し、バスの使用を許可するターゲットユニットに与えることにより、制御命令を与えられた該当ユニットだけが共通バスを使用可能としている。

なお、図 1 では、マイクロスケジューラ 24 は共通バス VBUS とは独立した制御バスにより各ネットワーク部 6 に指令を与えているが、図 3 のように共通バス VBUS を使用して各ネットワーク部 6 に指令を与えるようにしても構わない。

ただし、図 1 の構成の方がより高いバスの使用率を得ることができる。

【0051】

上記のマイクロスケジューラのプログラムは、該中央制御装置 20 によりスケジューリングされ、マイクロスケジューラテーブル 22 に格納される。

マイクロスケジューラテーブル 22 は、SRAM などを用いて構成され、中央制御装置 20 から各ス

【0048】

SCSI controller 14 of storage unit section 4 which can give command from the central control system 20 which did aforementioned [kejuuringu ] on basis of regeneration request from user terminal 32 sequential houses data which reading , reads out data which is required from disk drive 12 to designated address of buffer memory 18.

【0049】

In same way central control system 20 decides scheduling which forwards data to terminal 32 in slot unit , gives command vis-a-vis ATM controller 16 of network section 6, as for buffer memory 18 to [maikurosukeyuura ] following control of 24,data , reading , network section 6 to [maikurosukeyuura ] following control of 24, takes in data which reads out.

【0050】

And, from network section 6 in network 30 in user terminal 32 of each client the addressee \*, continual data of constant amount is forwarded in every constant period .

Well, [maikurosukeyuura ] 24, [maikurosukeyuuteeburu ] following to program of [maikurosukeyuuru ] which is housed in22, controls use of common bus VBUS which connects buffer memory 18 and network section 6.

namely, [maikurosukeyuura ] as for 24, just corresponding unit which can give control command control command of predetermined format (Figure 4 reference) which includes operation command for target unit [maikurosukeyuuteeburu ] at a time one by giving to target unit which uses reading , bus grant from entry of table inside 22, hasdesignated common bus as usable .

Furthermore, with Figure 1 , [maikurosukeyuura ] as for 24 command is given to each network section 6 common bus VBUS with control bus which becomesindependent, but like Figure 3 using common bus VBUS, it takes part the command in each network section 6 requiring, it does not care.

However, usage of bus where configuration of Figure 1 is highercan be acquired.

【0051】

program of above-mentioned [maikurosukeyuuru ] scheduling is done by said central control system 20, [maikurosukeyuuteeburu ]is housed in 22.

[maikurosukeyuuteeburu ] 22 configuration is done making use of SRAM , etc corresponds toeach slot from central

ロットに対応して書き換えられる。

これはディスク装置 12 の制御がスロット毎に変更されるのに対応している。

[0052]

なお、マイクロスケジュールテーブル 22 を複数のメモリバンクから構成し、あるバンクへの中央制御装置 20 からの書き込みと、他のバンクからのマイクロスケジューラ 24 による読み出しとを並列的に実行することが可能である。

[0053]

次に、マイクロスケジュールテーブル 22 内にマイクロスケジュールテーブルとして保持されるプログラムのフォーマットについて説明する。

マイクロスケジュールテーブル 22 は例えば図 4 のようなフォーマットの制御命令をテーブル形式で保持している。

図 4(a)のように、1 つのエントリには、少なくとも、命令と読み出し先アドレス(ソース・アドレス)と書き込み先アドレス(ディスティネーション・アドレス)の指令が書き込まれる。

本実施形態では詳しくは後述するが図 4(b)のように、さらにそのエントリを繰り返す回数の指令を付加して格納する。

[0054]

「命令」を表すフィールドにはバッファ 18 のメモリコントローラ(図示せず)と、ネットワーク部 6 の ATM コントローラ 16 への命令が格納される。

本実施形態では、この命令として次の 2 種類がある。

(1)送出命令:バッファメモリから ATM コントローラの packet メモリへの転送

(2)読み込み命令:ATM コントローラからバッファメモリへの転送

その他にも、ストレージシステムに RAID を採用した場合には、メモリコントローラへ RAID の演算を指示する命令をマイクロスケジューラから送ることが可能である。

この場合、中央制御装置がディスク装置の故障を検出し、マイクロスケジュールテーブルに RAID のための命令を書き込む。

[0055]

「送出命令」の場合、読み出し先アドレスはデータの格納されているバッファメモリ 18 の読み出しの先頭アドレスであり、書き込み先アドレスはタ

control system 20 and is rewritten.

This corresponds although control of disk drive 12 is modified every slot .

[0052]

Furthermore, [maikurosukeyuuruteeburu ] 22 configuration is done from memory bank of plural , the writing from central control system 20 to a certain bank and, it is possible with [maikurosukeyuura ] 24 from other bank to execute reading in arrayed .

[0053]

Next, you explain [maikurosukeyuuruteeburu ] [maikurosukeyuuruteeburu ] as concerning format of program which is kept inside 22.

[maikurosukeyuuruteeburu ] 22 has kept control command of format like for example Figure 4 with table form .

Like Figure 4 (a) , at least, command and reading destination address (source \*address ) with command of writing destination address ( [disutineeeshon ] \* address ) is written in entry of one .

With this embodiment it mentions later details, but like Figure 4 (b) , furthermore adding command of number of times which repeats entry , it houses .

[0054]

memory controller of buffer 18 (not shown ) with, command to ATM controller 16 of network section 6 is housed in field which displays "command " .

With this embodiment , there are next 2 kinds as this command .

From (1) forwarding command :buffer memory transfer to packet memory of ATM controller

(2) It reads, from command :ATM controller transfer to buffer memory

Even in addition, when RAID is adopted for storage unit system , it is possible to memory controller to send command which calculates RAID the display from [maikurosukeyuura ] .

In case of this , central control system detects breakdown of disk drive ,writes command for RAID in [maikurosukeyuuruteeburu ] .

[0055]

In case of "Forwarding command " , as for reading destination address with start address of reading of the buffer memory 18 where data is housed, writing destination address becomes

ターゲットユニットとなる ATM コントローラ 16 内の  
パケットメモリ 164 のアドレスの情報となる。

「読み込み命令」の場合は、その逆となる。

[0056]

図 5 に、マイクロスケジュールテーブルのフォー  
マットの一例を示す。

マイクロスケジュールテーブルは図 4 に示したマ  
イクロスケジューラの命令を複数個並べた形で  
構成されている。

マイクロスケジューラ 24 は、これを 1 エントリづ  
つ順番に実行していく。

[0057]

例えば、マイクロスケジューラ 24 内に次に実行  
するマイクロスケジュールテーブルのエントリ位  
置を保持するポインタカウンタを設け、マイク  
ロスケジューラ 24 はポインタカウンタの指し示す  
位置のエントリの内容をもとにバッファメモリ 18  
(のメモリコントローラ)とネットワーク部 6(の AT  
M コントローラ 16)にデータ転送の指令を出し、  
データの格納されているバッファメモリ 18 からタ  
ーゲットユニットとなるネットワーク部 6 ヘデータを  
送り出させ、あるいはターゲットユニットとなる  
ネットワーク部 6 からバッファメモリ 18 にデー  
タを読み込ませる。

[0058]

ところで、図 5 では、マイクロスケジュールテー  
ブルの各エントリに繰り返し回数のフィールドを設  
けている。

繰り返し回数のフィールドは各々のエントリーを  
何回実行してから次のエントリーに進むかの情  
報が示されている。

従って、同一のエントリ位置の命令を繰り返して  
いる間は、ポインタカウンタの値は更新されない  
ようにする。

[0059]

なお、詳しくは後述するが、図 7 のようにマイク  
ロスケジュールテーブルを複数のバンクに分割し、  
バンクを切替えながら実行することも可能で  
ある。

前述のように本実施形態では、中央制御装置 2  
0 のスケジューラではディスクアクセスのスケジ  
ューリングを行なうと同時にマイクロスケジュー  
ルテーブルを作成する。

このマイクロスケジュールテーブルはスロットに  
比べて非常に細かい単位で制御される。

data of the address of packet memory 164 inside ATM  
controller 16 which becomes target unit .

In case of "Reading command " , it becomes opposite.

[0056]

In Figure 5 , one example of format of  
[maikurosukeyuuruteeburu ] is shown.

[maikurosukeyuuruteeburu ] configuration is done in form  
which plurality arranges the command of [maikurosukeyuura ]  
which is shown in Figure 4 .

[maikurosukeyuura ] This at a time 1 entry it executes 24, in  
sequence .

[0057]

for example [maikurosukeyuura ] pointer counter which keeps  
entry position of [maikurosukeyuuruteeburu ] which  
isexecuted next inside 24 is provided, [maikurosukeyuura ] 24  
content of entry of position which pointer counter indicates  
buffer memory 18 (memory controller ) with puts out the  
command of data transfer to network section 6 (ATM  
controller 16 ) on basis of, sendingout data to network section  
6 which becomes target unit from the buffer memory 18  
where data is housed, Or from network section 6 which  
becomes target unit data is madeto read to buffer memory 18.

[0058]

By way, with Figure 5 , field of number of repetitions is  
provided in each entry of [maikurosukeyuuruteeburu ] .

After field of number of repetitions what time executing each  
entry , data whether it advances to following entry of, is  
shown.

Therefore, repeatedly while being, value of pointer counter  
requires is notrenewed command of same entry position .

[0059]

Furthermore, it mentions later details, but like Figure 7 also it  
ispossible to divide [maikurosukeyuuruteeburu ] into bank of  
plural , changeover toexecute bank .

Aforementioned way with this embodiment , when with  
[sukejuura ] of central control system 20 scheduling of disk  
access is done, [maikurosukeyuuruteeburu ] is drawn up  
simultaneously.

this [maikurosukeyuuruteeburu ] in unusual is controlled with  
small unit incomparison with slot .

例えば、本実施形態では 4word(16byte)を制御の最小単位とし、このデータを 4 クロックで読み出すものとしている。

この最小単位はバッファメモリの連続読み出しの単位(例えば SynchronousDRAM のバーストサイズ)とすると効率が良い。

最小単位は、以降、マイクロロットと呼ぶ。

従って、マイクロスケジューラ 24 は、マイクロロット毎にマイクロスケジューラテーブルから情報を読みだし、これを実行していくことになる。

[0060]

ところで、マイクロスケジューラテーブルを 1 スロット分(例えば 60msec)用意すると 1 クロックが 40nsec として約 380K エントリが必要になり、マイクロスケジューラテーブル用のメモリコストも中央制御装置 20 からの転送時間も問題になる。

そこで、次の方法によりエントリの数を抑えてマイクロスケジューラテーブル用のメモリコストを圧縮することができる。

[0061]

第 1 番目の方法は、マイクロスケジューラテーブルをスロットの分割数分繰り返して使用する方法である。

例えば 1K エントリのメモリを用意した場合はこれを 380 回繰り返し使用するものである。

これにより、メモリの容量を削減することができ、上記の例では 1/380 に減少する。

この方法は、マイクロスケジューラ 24 内にカウンタを設けるなどするだけで容易に実現可能である。

[0062]

第 2 番目の方法は、図 5 などに示したようにマイクロスケジューラテーブルのエントリに繰り返し回数を示すフィールドを設け、一つのエントリを何回か実行することである。

[0063]

また、以上の 2 つの方法を併用すれば、マイクロスケジューラテーブルのメモリ容量を効果的に減らすことができる。

ここで、本実施形態におけるマイクロロット、ミニロット、スロットの関係について説明する。

[0064]

With for example this embodiment it designates 4 word (16 byte ) as minimum unit of control, readout this data with 4 clock .

As for this minimum unit when unit of continual reading of buffer memory (burst size of for example SynchronousDRAM )with it does, efficiency is good.

minimum unit , later, calls Micros lot .

Therefore, [maikurosukeyuura ] 24 in every Micros lot starts reading data from [maikurosukeyuuteeburu ],to execute this means.

[0060]

When by way, [maikurosukeyuuteeburu ] is prepared equivalent of 1 slot (for example 60msec ), 1 clock approximately 380 Kentry become necessary as 40 nsec , [maikurosukeyuuteeburu ] memory cost of business and transfer time from central control system 20 become problem .

Then, holding down quantity of entry with following method , [maikurosukeyuuteeburu ]it can compress memory cost of business.

[0061]

method of first is method which number of divisions amount of slot uses [maikurosukeyuuteeburu ] over again.

When memory of for example 1Kentry is prepared, this it is something which380 repetitive use is done.

Because of this, it is possible, with above-mentioned exampledecreases to 1/380 to reduce capacity of memory .

[maikurosukeyuura ] counter such as is provided just does this method , is realizable easily inside 24.

[0062]

It is method of second , as shown in Figure 5 etc, to provide field which shows number of repetitions in entry of [maikurosukeyuuteeburu ], something timeto execute entry of one .

[0063]

In addition, if 2 method above are jointly used, it is possible todecrease memory capacity of [maikurosukeyuuteeburu ] in effective .

Here, you explain Micros lot , in this embodiment [minisurotto ], concerning therelationship of slot .

[0064]



図 6 は、マイクロロット、ミニロット、ロットの関係を示した図である。

まず、マイクロロットは前述のように一定のバスサイクルから構成され、この例では 10 サイクルでマイクロロットを構成している。

一つのマイクロロットはマイクロスケジュールテーブルの一つの命令に対応しているので、図 6 の拡大したマイクロロットではユーザ a へのデータが送られている。

[0065]

マイクロロットが一定の数だけ集まってミニロットが構成される。

図 6 にはユーザ a へのデータ送とユーザ c へのデータ送だけを示している。

この例では、ユーザ a へのデータは 1 ミニロット内に 9 マイクロロットを占めている。

このように、ミニロット内で同一転送命令を複数マイクロロット連続して実行させる手段は、前述の第 2 番目の方法により容易に実現可能である。

[0066]

ロットは一定の数のミニロットで構成される。

この例では 10 ミニロットで 1 ロットとなっている。

ミニロット 1 回は、前述の第 1 番目の方法によればマイクロスケジュールテーブルの一巡である。

このミニロットが繰り返し実行されるので、図 6 で示すように、ロットよりも細かい単位で一定の間隔でネットワーク部 6 に連続データが送り込まれ、1 ロット内における各ユーザ端末へのデータの転送が一定のレートで行われることを保証することが可能となる。

また、データの転送が一定のレートで行われることが保証されると、ATM コントローラ 16 内部の packets 164 の量を少なくすることが可能である。

[0067]

ロットは一定の数のミニロットで構成される。

この例では 10 ミニロットで 1 ロットとなっている。

ミニロット 1 回は、前述の第 1 番目の方法によ

Figure 6 Micros lot , [minisurotto ] , is figure which shows relationship of slot .

First, Micros lot aforementioned way configuration is done from the fixed bus cycle , with this example configuration does Micros lot with 10 cycle .

Because Micros lot of one corresponds to command of one of [maikurosukejuuruteeburu ] , with Micros lot which Figure 6 expands data to the user a is sent .

[0065]

Micros lot getting together , equal to fixed number [minisurotto ] is done the configuration .

Just data transmission to user a and data transmission to user c have been shown in Figure 6 .

With this example , as for data to user a 9 Micros lot are occupied inside 1 [minisurotto ] .

this way , plural Micros lot continuing same transfer command inside the [minisurotto ] , means which it executes is realizable easily with method of aforementioned second .

[0066]

slot configuration is done with [minisurotto ] of fixed number .

With this example it has become 1 slot with 10 [minisurotto ] .

[minisurotto ] one time according to method of aforementioned first is round of [maikurosukejuuruteeburu ] .

Because this [minisurotto ] repeatedly is executed , as shown with Figure 6 , continual data is sent by network section 6 with fixed spacing of small unit in comparison with slot , it becomes possible to guarantee that transfer of data to each user terminal inside 1 slot is done with fixed rate .

In addition , when it is guaranteed , that transfer of data is done with fixed rate , it is possible to decrease quantity of the packet memory 164 of ATM controller 16 interior .

[0067]

slot configuration is done with [minisurotto ] of fixed number .

With this example it has become 1 slot with 10 [minisurotto ] .

[minisurotto ] one time according to method of

ればマイクロスケジュールテーブルの一巡である。

このミニスロットが繰り返し実行されるので、図6で示すようにスロット内で各ユーザ端末へのデータの転送が一定のレートで行われることを保証することができる。

データの転送が一定のレートで行われることが保証されると、ATMコントローラ16内部のバッファメモリ164やユーザ端末32のバッファの容量を少なくすることができる。

[0068]

言い換えると、本実施形態によれば、スロットを複数に等分に分割した各ミニスロット内の一定の位置に、各ユーザ端末宛に送り出されるべきデータをネットワーク部6に取り込むための共通バスの使用が割り当てられるように、マイクロスケジュールのプログラムを作成することができ、このプログラムに従ってバス使用の制御を行うことにより、スロットよりも細かい単位の一定の間隔で、ネットワーク部6に連続データが送り込まれることを保証することができる。

このようなバス使用の制御は、前述の第1番目の方法と第2番目の方法を併用することにより、容易に実現可能である。

[0069]

また、各ユーザ端末宛に送り出されるべきデータのビットレートに応じた数の、スロットを複数に等分に分割したマイクロスロットが、該データをネットワーク部6に取り込むための共通バスの使用期間として割り当てられるように、プログラムを作成し、このプログラムに従ってバス使用の制御を行うことにより、各ユーザ端末の要求ビットレートに応じたバス使用期間を、各スロット内で確保することができる。

これを実現する1つの方法は、前記のようなエントリ内の繰り返し回数に従って実行を繰り返すことにより、ビットレートに応じた使用期間を割り当てるものである。

また、図6のようにミニスロットを併用して実現すると、より効果的である。

[0070]

次に、マイクロスケジュールテーブルを複数のバンクに分割する例について説明する。

図7は、マイクロスケジュールテーブルを2つのバンクに分割した例である。

これは図1で示すようにネットワーク部6が2つ

forementioned first isround of [maikurosukeyuuruteeburu ] .

Because this [minisurotto ] repeatedly is executed, as shown with Figure 6 ,you can guarantee that inside slot transfer of data to each user terminal is done with fixed rate .

When it is guaranteed, that transfer of data is done with thefixed rate , packet memory 164 of ATM controller 16interior and capacity of buffer of the user terminal 32 can be made little.

[0068]

paraphrase \* with, according to this embodiment , slot in fixed position inside each [minisurotto ] which is divided into equal parts in plural , inorder to be able to allot use of common bus in order to take in the data which it should you send out to each user terminal address to network section 6, to draw up program of [maikurosukeyuuru ] , it to be possible Following to this program , with fixed spacing of small unit , youcan guarantee that continual data is sent to network section 6 bycontrolling bus use, in comparison with slot .

Control of bus use a this way is realizable easily with the method of aforementioned first and jointly using method of the second .

[0069]

In addition, slot of a quantity which responds to bit rate of the data which it should you send out to each user terminal address Micros lot which is divided into equal parts in plural , in order to be allotted,as use period of common bus in order to take in said data to network section 6 in drawing up program , following to this program andcontrolling bus use depending, bus use period which responds to request bit rate of each user terminal , can be guaranteedinside each slot .

method of one which actualizes this, aforementioned wayfollowing to number of repetitions inside entry , is something which allots the use period which responds to bit rate by repeating execution.

In addition, like Figure 6 jointly using [minisurotto ] , it actualizes when,from, it is a effective .

[0070]

Next, you explain concerning example which divides [maikurosukeyuuruteeburu ] into the bank of plural .

Figure 7 is example which divides [maikurosukeyuuruteeburu ] into 2 bank .

This corresponds as shown with Figure 1 , when network

ある場合に対応している。

各々のバンクは各々のネットワーク部とのマイクロスケジュールに対応している。

【0071】

マイクロスケジューラ 24 は、マイクロスロット毎に Bank0 と Bank1 を交互に実行する。

図 8 に共通バス VBUS から 2 つのネットワーク部(0)と(1)への転送のタイミングを示す。

【0072】

この方法を使用することによる効果は、ATM コントローラ 16 のパケットメモリ部 164 のバンド幅ネックを緩和することにある。

例えば、ネットワーク部(0)に連続して要求が行なわれるとネットワーク部(0)のパケットメモリ 164 は毎クロック書き込みが発生することになり、SAR CHIP165 からの要求を受け付けられなくなって ATM パケットの送り出しに支障を来たすかもしれない。

例えば図 9 に示すように VBUSIF163 からのデータ転送により SAR CHIP165 はパケットメモリ 164 にアクセスできない。

パケットメモリ 164 をデュアルポートメモリで構成することも考えられるがコストの増大を招いてしまう。

【0073】

そこで、マイクロスケジューラ 24 からの命令を本実施形態のように 2 つに分けて 2 つのネットワーク部(0)と(1)に交互に実行させるようにすれば、パケットメモリ 164 へのバッファメモリ 18 からの転送は半分になり、バンド幅ネックが緩和される。

【0074】

次に、マイクロスケジューラテーブルとバンクとマイクロスロットの関係について説明する。

図 10 は本実施形態で説明する 2 つのバンクとマイクロスロットの関係を表したものである。

図 10 の一目盛は 1 マイクロスロットで ATM コントローラ(0)への送り出しは BANK0 に、ATM コントローラ(1)への送り出しは BANK1 に記述されている。

図 10 で小文字のアルファベットがそれぞれ異なったユーザ端末への送り出しを示している。

ATM コントローラ(0)へは a,a,a,b,b,c,c,c,c の順で送りだし、ATM コントローラ(1)へは e,f,g,h,i,j,j の

section 6 is a two .

Each bank corresponds to [maikurosukeyuuru ] of each network section.

【0071】

[maikurosukeyuura ] 24 executes Bank0 and Bank1 alternately in every Micros lot .

In Figure 8 2 network sections (0) with timing of transfer to (1) is shown from common bus VBUS.

【0072】

Effect is to ease dope width neck of packet memory section 164 of ATM controller 16 by fact that this method is used.

Continuing in for example network section (0), when request is done, every packet memory 164 of network section (0) decides that clock writing occurs, stops being accepted request from SARCHIP165 to feed of ATM packet the hindrance causes might.

As shown in for example Figure 9 , access it cannot designate SARCHIP165 as the packet memory 164 with data transfer from VBUSIF163.

configuration it can think of packet memory 164 with dual port memory that it does, but increase of cost is caused.

【0073】

Then, [maikurosukeyuura ] like this embodiment dividing command from 24 into two , if 2 network sections (0) with it executes it requires alternately in the(1), transfer from buffer memory 18 to packet memory 164 becomes half , dope width neck is eased.

【0074】

Next, you explain [maikurosukeyuuru teeburu ] with concerning relationship of bank between Micros lot .

Figure 10 is 2 bank which are explained with this embodiment and something which displays relationship of Micros lot .

As for one scale of Figure 10 as for feed to ATM controller (0) in the BANK0, as for feed to ATM controller (1) it is described to BANK1 with 1 Micros lot .

feed to user terminal where alphabet of lower case differs respectively in Figure 10 has been shown.

It starts sending to ATM controller (0) in order of a, a, a, b, b, c, c, c, c, c, to ATM controller (1) to order of e, f, g, h, i, j, j, j,

jj の順に送り出している。

この例のように共通バス VBUS では ATM コントローラ(0)と ATM コントローラ(1)へのデータがマイクロロット毎にインタリーブして使用されている。

[0075]

実施形態では二つのバンクに分けた場合を説明したが、これが 3 または 4 つ以上のバンクに別れている構成もネットワーク部および ATM コントローラとの関係で可能なことはいうまでもない。

[0076]

ところで、マイクロスケジュールテーブルをスロットの分割数分繰り返して使用し、および/または図 5 や図 7 のように各エントリの繰り返し回数を指令し、この情報に従って各エントリを繰り返し実行する場合、エントリのソースアドレスおよびディストネーションアドレスのフィールドは繰り返し実行される度に更新される。

例えば、マイクロスケジュールテーブルのエントリを一回実行すると 16byte のデータが転送される場合は 16byte のインクリメントが行なわれる。

このアドレスの更新機能は、マイクロスケジューラ 24 が DMA コントローラとして動作していることを示している。

一般に DMA コントローラはハードウェア資源の制約により、限られたチャンネル数のみをサポートしている。

例えば 2 チャンネルの DMA コントローラは、二つの異なるソース/ディストネーションのアドレスペアに対して転送をすることができるが、2 チャンネル以上の転送が必要な場合、プロセッサにより DMA の資源をセッティ直して転送することが必要になる。

[0077]

一方、本実施形態のような連続データサーバ装置を考えた場合、バッファメモリとネットワーク部のパケットメモリへの転送は、連続データサーバ装置のサポートするユーザの数だけチャンネル数が必要で、プロセッサが DMA をセッティ直すのはプロセッサにとってかなりの負担になる。

[0078]

ここで、マイクロスケジューラ 24 を DMA コントローラとして見ると、サポート可能なチャンネルの

j has sent out.

Like this example with common bus VBUS ATM controller (0) with data to the ATM controller (1) interleave making every Micros lot , it is used.

[0075]

With embodiment case where you divided into bank of two was explained, but this 3 or 4 configuration which has divided into the bank above as for possible thing does not have necessity to sayin network section or connection with with ATM controller .

[0076]

By way, number of divisions amount of slot you use [maikurosukejuuruteeburu ], over again like and/or Figure 5 and Figure 7 command you do number of repetitions of each entry ,follow to this data and when each entry repeatedly is executed, the source address of entry and field of [disutoneeshonadoresu ] are renewed to degreewhich repeatedly is executed.

When entry of for example [maikurosukejuuruteeburu ] is executed one time, when data of 16 byte is transferred, increment of 16 byte is done.

Renewal function of this address [maikurosukejuura ] 24 has shown fact that itoperates as DMA controller .

Generally DMA controller support has done only number of channels which is limitedby constraint of hardware resource .

DMA controller of for example 2channel can do transfer vis-a-vis address pair of the source / [disutoneeshon ] where two differs, but when transfer of 2 channel or moreis necessary, set doing again to do resource of DMA with processor , it becomes necessary to transfer.

[0077]

On one hand, when of continual data server device like this embodiment was thought,as for transfer to packet memory of buffer memory and network section, the support of continual data server device equal to quantity of user which isdone quantity of channel being necessary, as for those where the processor set does again to do DMA it becomes considerableburden for processor .

[0078]

When here, [maikurosukejuura ] you look at 24, as DMA controller because quantity of support possible channel means

数はマイクロスケジュールテーブルのエントリの数で制限されることになるので、マイクロスケジュールテーブルがメモリで実現されているために、例えば 1000 チャンネル以上のサポートでも可能になる。

マイクロスケジューラ 24 を使えば、DMA のためのセットが高々 1 スロットに 1 回で良く、スロット内では DMA のセットし直しをするためにプロセッサが割り込みで処理の中断をされることがない。

[0079]

次に、ユーザ端末またはコンテンツの入ったアーカイブ装置からデータをアップロードする場合について説明する。

アップロードは、連続データサーバ装置に新たなコンテンツを加える場合に必要な操作である。

アップロードは、例えば ATM 網 30 にアーカイブ装置から連続データを ATM パケットとして送り出しネットワーク部 6 から読み込む方法や、ネットワーク部 6 に直接テープデバイスあるいはディスクデバイス等を接続して読み込む方法などにより、共通バス VBUS にデータを流し込むものである。

[0080]

アップロードにおいても、読み込まれたデータは同様にマイクロスケジューラ 24 により確定的に制御されるタイミングで共通バス VBUS に送り出され、共通バス VBUS のデータはバッファメモリ 18 に書き込まれ、その後バッファメモリ 18 からストレージ部 4 に書き込まれる。

[0081]

ATM 網から受けたデータは、SAR CHIP165 によりパケットメモリ 164 に書き込まれる。

パケットメモリ 164 は、ATM 網 30 と連続データサーバ装置の間のバッファとして働く。

パケットメモリ 184 内部は、例えばリング状のバッファとして管理され、ATM 網 30 から書き込むとライトポインタ(Write Pointer)が進み、リードポインタ(Read Pointer)の方はスロット毎にマイクロスケジュールテーブルで設定した分だけネットワークプロセッサ 162 によって進められる。

通常はライトポインタ(Write Pointer)の管理は SAR CHIP165 によって行なわれる。

[0082]

to be restricted at quantity of entry of [maikurosukejuuruteeburu ], because [maikurosukejuuruteeburu ] is actualized with memory , even with the support above for example 1000channel it becomes possible.

If [maikurosukejuura ] 24 is used, set for DMA at highest is good to 1 slot being one time , inside slot set of DMA to dodoes again times when in order to do processor is discontinuedtreatment with interrupt are not.

[0079]

When next, data upload is done from archive device where user terminal or content enters, being attached, you explain.

upload , when new content is added to continual data server device , isnecessary operation.

As for upload , method of reading to for example ATM network 30 from feed network section 6 with continual data as ATM packet from archive device . Directly connecting tape device or disk device etc to network section 6, the data sink it is something which is packed in common bus VBUS with the method etc which it reads.

[0080]

Regarding upload , data which is read is sent out by common bus VBUS with timing which is controlled decide by [maikurosukejuura ] 24 in same way,data of common bus VBUS is written by buffer memory 18, after that from the buffer memory 18 is written to storage unit section 4.

[0081]

data which is received from ATM network is written to packet memory 164 by SARCHIP165.

packet memory 164 works as buffer between ATM network 30 and continual data server device .

When packet memory 184interior is managed writes from ATM network 30 as buffer of the for example ring , write pointer (WritePointer ) advances, lead pointer (ReadPointer ) equal to amount whichis set every slot with [maikurosukejuuruteeburu ] is advanced with network processor 162.

As for management of write pointer (WritePointer ) it is done usually with SARCHIP165.

[0082]

マイクロスケジューラテーブルには、ATM コントローラ 16 のパケットメモリ 164 からバッファメモリ 18 に書き込む命令を書いておく。

この命令により ATM コントローラ 16 のパケットメモリ 164 からは共通バス VBUS にデータが送られ、バッファメモリ 18 は共通バス VBUS 上のアップロードデータをバッファメモリ 18 に書き込む。

[0083]

この場合、もし ATM 網 30 からのパケットメモリ 164 へのデータ供給がマイクロスケジューラ 24 で吸い上げる速度より遅ければ、バッファ領域のリングバッファが empty になってしまうかもしれない。

もし供給が追いつかずに empty になると、マイクロスケジューラ 24 からの指示に対応するデータが共通バス上に存在しない状態が発生することになるので、ネットワーク部 6 のネットワークプロセッサ 162 が empty 状態を検出し、マイクロスケジューラ 24 がパケットメモリ 164 の読み出しを要求するタイミングで制御信号を出して、共通バス VBUS にはデータを乗せないようにする。

また、データが共通バス VBUS に乗っていないので、マイクロスケジューラ 24 はエントリ中のアドレスをインクリメントせず、バッファメモリ 18 はメモリに書き込みを行なわないよう指示する。

[0084]

上記の条件は、ATM 網 30 から来るビットレートが一定でない場合で、ビットレートが一定の場合には上記の機構を働かせなくてもパケットメモリ 164 がバッファとして働き、パケットメモリ 164 が empty になることはない。

一方、ビットレートが一定でない場合はマイクロスケジューラ 24 には予想される最大のレートで ATM 網 30 からの読み出しの命令を入れておき、前述した機能を働かせると良い。

[0085]

以上、一例として、ATM 網から VBUS 経由でアップロードデータ書き込む方法を示したが、VBUS に直接データを供給するデバイスをつけることが可能であることはいうまでもない。

[0086]

次に、マイクロスケジューラ 24 およびマイクロスケジューラテーブル 22 の構成について説明する。

command which from packet memory 164 of ATM controller 16 is written to buffer memory 18 is written to [maikurosukejuuruteeburu ].

data is sent by common bus VBUS from packet memory 164 of ATM controller 16 by the this command , buffer memory 18 writes upload data on common bus VBUS to buffer memory 18.

[0083]

In case of this , if it is slower than speed which data supply to packet memory 164 from ATM network 30 sucks up with [maikurosukejuura ] 24 , ring buffer of buffer region becomes empty , might.

Supply without overtaking, when it becomes empty, because [maikurosukejuura ] it means that state where data which corresponds to display from 24 does not exist on common bus occurs, network processor 162 of network section 6 detects empty state , [maikurosukejuura ] control signal is put out with timing to which 24 requires reading of packet memory 164, data is not placed to the common bus VBUS, requires.

In addition, because data is not riding in common bus VBUS, [maikurosukejuura ] as for 24 address in entry increment it does not do, in order not to do writing in memory , display it does buffer memory 18.

[0084]

Above-mentioned condition when with when bit rate which comes from ATM network 30 is not fixed, bit rate is fixed not being able to use the above-mentioned mechanism , packet memory 164 it works as buffer , there are not times when packet memory 164 becomes empty.

On one hand, when bit rate is not fixed, in [maikurosukejuura ] 24 command of the reading from ATM network 30 is inserted with maximum rate which is expected, when function which is mentioned earlier is used it is good.

[0085]

Above, as one example , from ATM network upload data is written method which was shown with VBUS going by way of, but as for being possible to attach device which directly supplies data to VBUS it is not necessary to say.

[0086]

Next, you explain [maikurosukejuura ] 24 and [maikurosukejuuruteeburu ] concerning configuration of 22.

図 11 に、マイクロスケジューラ 24 の内部構成およびマイクロスケジューラテーブル 22 の構成の一例を示す。

[0087]

マイクロスケジューラテーブル 22 には SRAM を使用し、中央制御装置 20 から書き込めるようになっている。

図 11 の構成では、SRAM22 がネットワーク部 16 の数に合わせて二つのバンク Bank0 と Bank1 分けられており(図 7 参照)、各メモリバンクにはそれぞれ対応するターゲットユニットへの指示の基となるプログラムが格納される。

これに対応して、後述するアドレスカウンタおよび繰り返しカウンタとも、2 組づつ存在する。

これらを切替えるのは図 11 中に示した BANK 信号である。

BANK 信号は、1 マイクロロットごとに 1 と 0 の反転を繰り返す信号である。

これらによって、ターゲットユニットのそれぞれに対する指示を 1 マイクロロットごとにインターリーブして転送できるようにしている。

[0088]

マイクロスケジューラテーブル 22 からマイクロスケジューラ 24 内に読み出されたデータは一旦、レジスタ 241 に蓄えられる。

これとともに、命令のフィールドの内容が「送出命令」の場合には、バッファメモリ 18 にソースアドレスと命令が、ATM コントローラ 16 にはデスティネーションアドレスと命令が各々転送され、「読み込み命令」の場合には、バッファメモリ 18 にデスティネーションアドレスと命令が、ATM コントローラ 16 にはソースアドレスと命令が各々転送される。

[0089]

ソースアドレスおよびデスティネーションアドレスは、それぞれインクリメント器 243,242 により、1 マイクロロットで転送されるデータ量に相当するバッファのアドレス分だけインクリメントされた後、書き込みゲート 244 によりマイクロスケジューラテーブル 22 に再び書き戻される。

[0090]

CONT ビットは、初期値が 1 にセットされ、このときレジスタ 241 の繰り返し回数データが繰り返しカウンタ 248 または 250 にロードされ、CONT ビットには 0 が書き戻される。

In Figure 11, [maikurosukeyuura ] internal configuration of 24 and [maikurosukeyuuteeburu ] one example of configuration of 22 is shown.

[0087]

[maikurosukeyuuteeburu ] You use SRAM to 22, can write from central control system 20 it groans.

With configuration of Figure 11, SRAM 22 adjusting to number of network sections 16, bank Bank0 and Bank1 min of two \* and others\* \* time (Figure 7 reference), program which becomes basis of display to the target unit which corresponds respectively is housed in each memory bank .

Corresponding to this, also address counter and repetition counter which it mentions later exist, at a time 2 sets .

Fact that these are changed is BANK signal which is shown in the Figure 11 .

BANK signal is signal which repeats inverting of 1 and 0 in every Micros lot .

With these, interleave designating display target unit for respectively as every Micros lot , it can transfer, it has required.

[0088]

[maikurosukeyuuteeburu ] data which [maikurosukeyuura ] reads out from 22 inside 24 once is stored in register 241.

When with this, content of field of command is "Forwarding command", in the buffer memory 18 source address and command, by ATM controller 16 [disutoneeshonadoresu ] with command each are transferred, when it is a "Reading command", in buffer memory 18 [disutoneeshonadoresu ] with command, in the ATM controller 16 source address and command each is transferred.

[0089]

[maikurosukeyuuteeburu ] You write source address and [disutoneeshonadoresu ], on 22 again equal to address amount of buffer which is suitable to data amount which is transferred with 1 Micros lot by respective increment vessel 243,242, increment after being done, with writing gate 244 and are reset.

[0090]

As for CONT bit, initial value set makes 1, at time of the this number of repetitions data of register 241 repeats and counter 248 or load makes 250, 0 writes on CONT bit and is reset.

繰り返しカウンタ 248 または 250 の内容は、該当エントリが実行される度に 1 づつデクリメントされ、繰り返しカウンタ 248 または 250 が 0 になったら、次に行うマイクロスケジューラテーブルのエントリ位置を指し示すポインタカウンタ 251 または 252 をインクリメントするとともに、再び CONT ビットに 1 が書き戻される。

[0091]

図 11 では、SRAM22 はデュアルポート構成で、中央制御装置 20 とマイクロスケジューラ 24 の両方からアクセスできるようになっている。

SRAM22 をデュアルポート構成ではなく、ダブルバッファ構成にすることも可能である。

この場合、中央制御装置 20 が一方のバンクに書き込んでいる間、もう一方のバンクをマイクロスケジューラが読み出す。

[0092]

マイクロスケジューラテーブルの繰り返し回数により、ビデオサーバ装置では異なるデータレートのビデオを混在させることが可能である。

例えば、VBUS のデータ転送速度を 33MHz 32 bit で 1056Mbit/sec であるとき、マイクロスケジューラのエントリ数が 4096 として繰り返し回数 1 では 1056Mbit/4096 = 258Kbit/sec となる。

4Mbit/sec の転送には繰り返し回数を 16 にする。

このように繰り返し回数を設定することで容易にビデオ転送レートを設定することが可能である。

[0093]

転送レートには 258Kbit の量子化誤差が存在することになるが、端末 32 や、ATM コントローラ 16 にバッファがあるので、この量子化誤差はスロット毎に繰り返し回数を調整することで一定の範囲内に収めることができる。

[0094]

マイクロスケジューラ 24 を使った場合の効果の一つは、高度にパイプライン化されたアクセスによるバスの効率的利用である。

マイクロスケジューラ 24 を用いた場合、図 12 に示されるようにバッファメモリ 18 からネットワーク部 6 に読み出されるデータのアクセス時間を隠蔽でき、かつデータを読み出しながらデータの転送が可能となる。

[0095]

When repetition counter 248 or content of 250 at a time 1 decrement is done in degree where corresponding entry is executed and the repetition counter 248 or 250 becomes 0, as pointer counter 251 which indicates the entry position of [maikurosukeyuuteeburu] which is executed next or 252 is done increment, 1 writes on CONT bit again and is reset.

[0091]

With Figure 11, SRAM 22 with dual port configuration, central control system 20 from [maikurosukeyuura] the access is possible both of 24, it groans.

SRAM 22 it is not a dual port configuration, also it is possible to make double buffer configuration.

In case of this, while central control system 20 is writing to bank of onside, [maikurosukeyuura] reads out bank of another.

[0092]

With number of repetitions of [maikurosukeyuuteeburu], with video server device video of data rate which differs it is possible to exist together.

When data transfer speed of for example VBUS being 1056 Mbit/sec with 33 MHz 32bit, the quantity of entry of [maikurosukeyuura] with number of repetitions 1 1056 MB / 4096 = 258Kbit / s ago as 4096.

In transfer of 4 Mbit/sec number of repetitions is designated as 16.

this way it is possible to set video transfer rate easily by fact that number of repetitions is set.

[0093]

In transfer rate it means that quantization error of 258 Kbit exists, but because there is a buffer in terminal 32 and ATM controller 16, as for this quantization error by the fact that number of repetitions is adjusted every slot it can supply inside fixed range.

[0094]

[maikurosukeyuura] one of effect when 24 was used high-level to pipeline is efficient utilization of bus with access which is converted.

When [maikurosukeyuura] 24 is used, as shown in Figure 12, access time of data which reads out from buffer memory 18 in network section 6 hiding it is possible, at same time data reading transfer of data becomes possible.

[0095]



通常の方式では、複数のリクエストの調停に時間がかかり、しかもデータの転送と読み出しを並行して行なえないので全ての処理が直列に行なわれるが、マイクロスケジューラ方式では図 12 のように並行して処理を行なうことでバスの使用効率が極めて高い。

【0096】

(実施形態 2)次に、実施形態 2 について説明する。

図 13 に本実施形態に係る連続データサーバ装置の構成を示す。

【0097】

本実施形態の連続データサーバ装置は、連続データを格納している所定台数のデータ記憶装置 12 と該データ記憶装置 12 からデータを読み出すデータ記憶制御装置 14 とからなる、複数(ここでは 4 系統とする)のストレージ部 4、該ストレージ部 4 から読み出したデータを一時記憶する、ストレージ部 4 に対応して設けられたバッファメモリ 18、該バッファメモリ 18 のデータを各端末 32 を宛先としてネットワーク 30 に送り出す複数(ここでは 2 台とする)のネットワーク部 6、ストレージ部 4 とバッファメモリ 18 とネットワーク部 6 を接続する共通バス VBUS、端末 32 からの要求に基づいてデータ記憶装置 12 からデータを読み出してネットワーク 30 に送り出すまでのスケジューリング、ネットワークの設定をはじめとして、システム全体の制御を行なう中央制御装置 20、該中央制御装置 20 により確定されたマイクロスケジューリングのプログラムを格納するマイクロスケジューリングテーブル 22、該プログラムに従ってネットワーク部 6 に共通バス VBUS を使用する権利を確定的に割り当てるマイクロスケジューラ 24 を備えている。

【0098】

すなわち、本実施形態は、実施形態 1 のストレージ部 4 およびバッファメモリ 18 を複数系統(図 13 では 4 系統)分、共通バス VBUS に接続し、各バッファメモリ 18 を各ネットワーク部 6 の ATM コントローラ 16 に接続したものである。

【0099】

本実施形態のように複数系列のストレージ部 4 を設けた場合、ディスク装置 12 にデータをストライピングして配置し、ディスクを並列に読み出すことにより、ディスクアクセスのバンド幅を大きくでき、より多くのユーザに同時に連続データをサービスすることができる。

With conventional system, time to be required for arrange of request of plural, furthermore because it cannot transfer data and reading in parallel, all treatment is done in series array, but with [maikurosukejuura] system like Figure 12 in parallel, efficiency in use of bus quite is high by fact that it treats.

【0096】

(embodiment 2) Next, you explain concerning embodiment 2.

configuration of continual data server device which relates to this embodiment in the Figure 13 is shown.

【0097】

Continual data server device of this embodiment consists of data storage controller 14 which read-out does data from data storage device 12 and said data storage device 12 of specified number of devices which houses continual data, storage unit section 4 of plural (Here 4 system it does), reading is the data is remembered at one time from said storage unit section 4, Corresponding to storage unit section 4, until reading \* it sends out the data to network 30 from data storage device 12 network section 6 of plural (Here 2 it does) which it sends out to network 30 data of buffer memory 18, said buffer memory 18 which is provided with each terminal 32 as addressee, storage unit section on basis of request from common bus VBUS, terminal 32 which connects 4 and buffer memory 18 and network section 6, with setting of scheduling, network as beginning, program of [maikurosukejuura] which is decided by central control system 20, said central control system 20 which controls the system entirety is housed [maikurosukejuuruteeburu] 22, following to said program and right common bus VBUS to be used for network section 6 decide is allotted, [maikurosukejuura] it has 24.

【0098】

It is something where namely, this embodiment, storage unit section plural system (With Figure 13 4 system) amount, connects 4 of embodiment 1 and buffer memory 18 to common bus VBUS, each buffer memory 18 connects to ATM controller 16 of each network section 6.

【0099】

Like this embodiment when storage unit section 4 of plural array is provided, [sutoraipingu], it arranges data in disk drive 12, dope width of disk access it can make large by reading out disk in parataxis, continual data the service is possible simultaneously to many user.

また、複数系列あるストレージ部 4 の一部をパリテイ符号として割り当てることにより、RAID システムを構成することができる。

[0100]

図 14 は、各バッファメモリ 18 からネットワーク部 6 にデータを送り出すときのマイクロスケジューラからの命令とバッファメモリ 18 からネットワーク部 6 に送り出されるデータの関係を示したものである。

タイミング 1 でマイクロスケジューラ 24 からバッファメモリ 18 の読み出し要求 1 が各バッファメモリ 18 に出されると、時刻 6 でバッファメモリ(0)からデータが出力され、時刻 7 でバッファメモリ(1)からデータが出力され、時刻 8 でバッファメモリ(2)からデータが出力され、時刻 9 でバッファメモリ(3)からデータが出力されるというように、順次 4 台のバッファメモリ 18 からデータが読み出されている。

ここで重要な点は、時刻 1 の読みだし要求 1 に対する応答が時刻 6 から始まっていることで、パイプライン動作によりメモリのレーテンシを隠蔽している。

[0101]

本実施形態ではマイクロスケジューラ 24 からバッファメモリ 18 および ATM コントローラ 16 への制御バスを専用に設けているが、実施形態 1 と同様に、マイクロスケジューラ 24 から専用のバスを設けずに、共通 VBUS を共用する方法もある。

[0102]

(実施形態 3)次に、実施形態 3 について説明する。

実施形態 2 では ATM コントローラ 16 とバッファメモリ 18 間のバスをマイクロスケジューラ 24 で制御していたが、マイクロスケジューラ 24 で SCSI コントローラ 14 とバッファメモリ 18 の間のバスを制御することもできる。

[0103]

図 15 に本実施形態に係る連続データサーバ装置の構成を示す。

本実施形態の連続サーバ装置は、基本的には実施形態 2 と同じ構成であるが、マイクロスケジューラ 24 がストレージ側とネットワーク側の両方についてバス使用の制御を行う点、マイクロスケジューラ 22 に格納するマイクロスケジューラのプログラムをストレージ側とネットワーク側の夫々について作成する点が相違する。

In addition, RAID system configuration is possible portion of storage unit section 4 which is plural array as parity code by allotting.

[0100]

Figure 14, when from each buffer memory 18 sending out data to network section 6, is something which shows relationship of data which from command and buffer memory 18 from [maikurosukeyuura] is sent out to network section 6.

When with timing 1 reading demand 1 for buffer memory 18 is put out to each buffer memory 18 from [maikurosukeyuura] 24, way with time 6 data is outputted from the buffer memory (0), with time 7 data is outputted from buffer memory (1), with time 8 data is outputted from buffer memory (2), with the time 9 data is outputted from buffer memory (3), data is read out from buffer memory 18 of sequential 4table.

time 1 it starts reading important point, here, by fact that response for request 1 has started from time 6, with pipeline operation [reetenshi] of memory hiding it does.

[0101]

With this embodiment from [maikurosukeyuura] 24 control bus to buffer memory 18 and ATM controller 16 is provided in dedicated, but in same way as embodiment 1, [maikurosukeyuura] without providing pass of dedicated from 24, there is also a method which shares common VBUS.

[0102]

(embodiment 3) Next, you explain concerning embodiment 3.

With embodiment 2 bus between ATM controller 16 and buffer memory 18 was controlled with [maikurosukeyuura] 24, but [maikurosukeyuura] it is possible also to control bus between SCSI controller 14 and buffer memory 18 with 24.

[0103]

configuration of continual data server device which relates to this embodiment in the Figure 15 is shown.

Continual server device of this embodiment is same configuration as embodiment 2 in the basic, but [maikurosukeyuura] 24 point which controls bus use concerning the both of storage unit side and network side, [maikurosukeyuuru] point which draws up the program of [maikurosukeyuuru] which is housed in 22 concerning respectively of storage unit side and network side

一方側の夫々について作成する点が相違する。

[0104]

本実施形態のように、マイクロスケジューラ 24 でストレージ側とネットワーク側の両方を制御する場合は、マイクロスケジューラ 24 およびマイクロスケジューラテーブル 22 の構成は次の 2 つの場合が考えられる。

[0105]

(1)図 16 で示すように、ストレージ側を制御するマイクロスケジューラ 24 およびマイクロスケジューラテーブル 22 とネットワーク側を制御するマイクロスケジューラ 24 およびマイクロスケジューラテーブル 22 を独立に構成する。

[0106]

(2)図 17 で示すように、ストレージ側を制御するマイクロスケジューラ 24 とネットワーク側を制御するマイクロスケジューラ 24 は共通で、マイクロスケジューラ管理テーブル 22 がストレージ側を制御する部分と、ネットワーク側を制御する部分の 2 つに分かれている。

[0107]

また、図 18 に示すように、SCSI コントローラ 14 の内部構成は、ネットワーク部 6 の ATM コントローラ 16 の内部構成(図 2)の SAR CHIP165 を SCSI CHIP185 に置き換え、パケットメモリ 164 をストレージメモリ 184 に置き換え、外部への ATM 網を SCSI に置き換えられたものとなる。

[0108]

本実施形態では、ネットワーク側の部分については、実施形態 1 あるいは実施形態 2 と同様であるが、ストレージ側については、ストレージ部 4 から読み出されたデータをバッファメモリ 18 に取り込む際の共通バスの使用を割り当てるマイクロスケジューラのプログラムを作成する。

[0109]

本実施形態では、まず、ディスク装置 12 から読み出されたデータは SCSI コントローラ 14 のストレージメモリ 184 に蓄積される。

SCSI コントローラ 14 のストレージメモリ 184 の容量は、接続されたディスク装置 12 に 1 スロットでアクセスされるデータサイズより大きい。

例えば、1 スロットで 64KByte アクセスされるディスク装置 12 が 1 スロットで 4 つに対して読み出し命令を SCSI コントローラ 14 から発行する場合は、ストレージメモリ 184 の容量は、256KByte とする。

differs.

[0104]

Like this embodiment, when [maikurosukeyuura] both of storage unit side and network side is controlled with 24, [maikurosukeyuura] 24 and [maikurosukeyuuruuteeburu] configuration of 22 is thought in next 2 cases.

[0105]

As shown with (1) Figure 16, storage unit side is controlled [maikurosukeyuura] 24 and controls [maikurosukeyuuruuteeburu] 22 and network side [maikurosukeyuura] configuration makes 24 and [maikurosukeyuuruuteeburu] 22 independent.

[0106]

As shown with (2) Figure 17, controls storage unit side [maikurosukeyuura] controls 24 and network side [maikurosukeyuura] 24 with common, has divided into two of the portion which controls portion and network side where [maikurosukeyuuru] administration table 22 controls storage unit side.

[0107]

In addition, as shown in Figure 18, internal configuration of SCSI controller 14 replaces SAR CHIP165 of internal configuration (Figure 2) of ATM controller 16 of network section 6 to the SCSI CHIP185, packet memory 164 replaces to storage unit memory 184, ATM network to outside replaces to SCSI and \* becomes landing net.

[0108]

With this embodiment, it is similar to embodiment 1 or embodiment 2 concerning the portion of network side, but case where data which reads out from storage unit section 4 concerning storage unit side, is taken in to buffer memory 18 program of [maikurosukeyuuru] which allots use of common bus is drawn up.

[0109]

With this embodiment, first, as for data which reads out from disk drive 12 compilation it makes storage unit memory 184 of SCSI controller 14.

capacity of storage unit memory 184 of SCSI controller 14 than data size which access is done is larger to disk drive 12 which is connected with 1 slot.

disk drive 12 which 64 KByte access is done being 1 slot with for example 1 slot, when reading command is issued from SCSI controller 14 vis-a-vis 4, capacity of storage unit memory 184 becomes 256 KByte.

なる。

【0110】

以下では、インプリメントが最も容易であるダブルバッファをストレージメモリ 184 に使用した場合について説明する。

あるスロットでストレージメモリ 184 に読み出されたデータは、次のスロットでバッファメモリ 18 に転送される。

ダブルバッファを使っているので、バッファメモリ 18 に転送している間、もう一方のバンクはディスク装置 12 からストレージメモリ 184 への転送を行っていることになる。

ストレージメモリ 184 とバッファメモリ 18 間の転送は一定の速度で、ほぼ 100%バスを使つての転送が可能である。

この場合、ストレージメモリ 184 はディスク装置 12 から一定の速度で出てこないデータをバッファリングする働きを行う。

【0111】

一方、前述のようにコンテンツをロードする場合、このバスは全く逆の方向に働く。

すなわち、バッファメモリ 18 からストレージメモリ 184 にマイクロスケジューラ 24 の指示によりデータが転送され、次にストレージメモリ 284 からディスク装置 12 にデータが転送される。

この場合にも、ストレージメモリ 184 はバッファとして働く。

【0112】

なお、本実施形態では、マイクロスケジューラ 24 がストレージ側だけについてバス使用の制御を行う実施形態も可能である。

この場合、上記構成において、マイクロスケジューラ 24 からネットワーク部 6 へ情報を伝える手段が不要となり、マイクロスケジューラ 24 およびマイクロスケジューラテーブル 22 はストレージ側対応に設けられ、中央制御装置 20 はストレージ側に対するマイクロスケジューリングのみ行うように修正する。

また、この場合、ストレージ側についての構成・動作は上記と同様であり、ネットワーク側についての構成・動作は、従来技術と同様になる(共通バスの使用権は調停装置により制御される)。

【0113】

【0110】

At below, when double buffer where implementation is easiest is used for the storage unit memory 184 being attached, you explain.

data which with a certain slot reads out in storage unit memory 184 is transferred to buffer memory 18 with following slot .

Because double buffer is used, while transferring to buffer memory 18, bank of another means to transfer to storage unit memory 184 from disk drive 12.

As for transfer between storage unit memory 184 and buffer memory 18 with fixed speed ,using almost 100% bus , transfer is possible.

In case of this , storage unit memory 184 does function which buffering does data which does not come out of disk drive 12 with fixed speed .

【0111】

On one hand, aforementioned way when content is done load ,this bus works completely in direction of opposite.

data is transferred from namely, buffer memory 18 [maikurosukejuura ] by display of 24 in the storage unit memory 184, next from storage unit memory 284 data is transferred to disk drive 12.

In case of this , storage unit memory 184 works as buffer .

【0112】

Furthermore, with this embodiment , [maikurosukejuura ] 24 also embodiment which controls bus use concerning just storage unit side is possible.

In case of this , [maikurosukejuura ] from 24 means which conveys data becomes unnecessary to network section 6 in above-mentioned configuration ,[maikurosukejuura ] 24 and [maikurosukejuuruteeburu ] 22 is provided in storage unit side correspondence, in orderonly [maikurosukejuuringu ] for storage unit side to do corrects central control system 20.

In addition, in case of this , as for configuration \* operation concerning storage unit side being similar to description above, configuration \* operation concerning network side becomes similar to Prior Art , (Use right of common bus is controlled by arrange device ).

【0113】

以上の各実施形態では SCSI を使用してディスクを接続しているが、ファイバーチャネル等他のディスクインタフェースが使用できるのはいうまでもない。

また、ネットワーク部に ATM を使用しているが、イーサネットなど他のネットワークを用いるのも好ましい実施形態の一例である。

また、以上の実施形態ではマイクロスケジューラはシステムに 1 個であったがバッファメモリ毎に設けることも可能である。

本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

[0114]

**[発明の効果]**

本発明(請求項 1)に係る計算機装置によれば、各ユニットからのバス使用要求に依存してバス使用権の調停を行うのではなく、格納手段に格納されたプログラムに従って割当手段により、各ユニットが共通バスを使用する権利を決定論的に割り当てるようにしたので、予め定めたスケジューリングによりバス使用権を割り当てるため、各ユニットのバス使用時期を確定的に保証することができる。

このため、メモリへの指示を先行発行することができるので、メモリのレーテンシを隠蔽することができる。

従って、バスの使用率の向上を図ることができる。

[0115]

本発明(請求項 9)に係る連続データサーバによれば、バッファメモリユニットと通信制御ユニットの間でのデータ転送に共通バスを使用する権利を、プログラムに従って各通信制御ユニットに対し決定論的に割り当てるようにしたので、請求項 1 の発明の効果に加え、一定の間隔で通信制御ユニットに連続データが送り込まれ、ひいては、一定の間隔で通信路に連続データが送り出されることを保証することができる。

従って、通信制御ユニットが通信路に連続データを送り出すまで該データを一時記憶するメモリの容量を少なくでき、また、通信路の先に存在して連続データを再生するユーザ端末において受信したパケットを保持するバッファの容量を少なくできる。

**[図面の簡単な説明]**

With each embodiment above using SCSI , you connect disk , but it is not as for other disk interface necessary such as fiber channel to say being able to use.

In addition, ATM is used for network section, but it is the other network also it is desirable a one example of embodiment whose such as Ethernet to use.

In addition, with embodiment above as for [maikurosukejuura ] 1 was in system ,but also it is possible to provide in every buffer memory .

this invention, it is not something which is limited in embodiment which the description above is done, various deforming in technological range , it can execute.

[0114]

**[Effects of the Invention ]**

According to computer device which relates to this invention (Claim 1 ), depending on the bus use request from each unit , not to be to arrangement bus use right ,following to program which is housed in storage means , it allots theright each unit to use common bus with allotment means , to the deterministic because it required, In order to allot bus use right with scheduling which is decided beforehand, bus use time of each unit can be guaranteeddecide.

Because of this , because it can precede can issue display to memory , [reetenshi ] of memory hiding is possible.

Therefore, it is possible to assure improvement of usage of bus .

[0115]

According to continual data server which relates to this invention (Claim 9 ),following right to use common bus for data transfer between buffer memory unit and communication control unit , to program , it allots to deterministic , vis-a-vis each communication control unit because it required, in addition to Effect of Invention of Claim 1 , withfixed spacing in communication control unit continual data sending, You can guarantee that continual data is sent out to communications line with consequent , fixed spacing .

Therefore, until communication control unit sends out continual data to communications line ,it can make capacity of buffer which keeps packet which itreceives capacity of memory which remembers said data at one timeit can make little, in addition, communications line existing first, in the user terminal which regeneration does continual data little.

**[Brief Explanation of the Drawing (s )]**

## 【図1】

本発明の実施形態 1 に係る連続データサーバ装置の構成を示す図

## [Figure 1 ]

Figure which shows configuration of continual data server device which relatesto embodiment 1 of this invention

## 【図2】

ネットワーク部の内部構成の一例を示す図

## [Figure 2 ]

Figure which shows one example of internal configuration of network section

## 【図3】

同実施形態に係る連続データサーバ装置の他の構成を示す図

## [Figure 3 ]

Figure which shows other configuration of continual data server device whichrelates to same embodiment

## 【図4】

マイクロスケジュールテーブル 22 に格納される単位プログラムのフォーマットの一例を示す図

## [Figure 4 ]

[maikurosukeyuuruteeburu ] Figure which shows one example of format of unit program which is housed in 22

## 【図5】

マイクロスケジュールテーブルのフォーマットの一例を示す図

## [Figure 5 ]

Figure which shows one example of format of [maikurosukeyuuruteeburu ]

## 【図6】

マイクロロット、ミニロット、ロットの関係を説明するための図

## [Figure 6 ]

Micros lot , [minisurotto ], figure in order to explain relationship of the slot

## 【図7】

マイクロスケジュールテーブルのフォーマットの他の例を示す図

## [Figure 7 ]

Figure which shows other example of format of [maikurosukeyuuruteeburu ]

## 【図8】

バンク切替えによる2つのネットワーク部への転送のタイミングを示すタイミングチャート

## [Figure 8 ]

timing chart which shows timing of transfer to 2 network sections with bank changeover

## 【図9】

ネットワーク部内でのデータの流れを説明するための図

## [Figure 9 ]

Figure in order to explain flow of data in network circles

## 【図10】

2つのバンクとマイクロロットの関係を説明するための図

## [Figure 10 ]

2 bank and figure in order to explain relationship of the Micros lot

## 【図11】

マイクロスケジュールの内部構成の一例およびマイクロスケジュールテーブルの構成の一例を示す図

## [Figure 11 ]

one example of internal configuration of [maikurosukeyuura ] and figure which shows one example of configuration of [maikurosukeyuuruteeburu ]

## 【図12】

同実施形態におけるパイプライン処理を説明するための図

## [Figure 12 ]

Figure in order to explain pipeline processing in same embodiment

## 【図13】

本発明の実施形態 2 に係る連続データサーバ装置の構成を示す図

## [Figure 13 ]

Figure which shows configuration of continual data server device which relatesto embodiment 2 of this invention

【図14】

各バッファメモリからネットワーク部にデータを送り出すときのマイクロスケジューラからの命令とバッファメモリからネットワーク部に送り出されるデータの関係を説明するための図

[Figure 14 ]

When from each buffer memory sending out data to network section, the figure in order to explain relationship of data which from command and buffer memory from [maikurosukejuura ] is sent out to network section

【図15】

本発明の実施形態 3 に係る連続データサーバ装置の構成を示す図

[Figure 15 ]

Figure which shows configuration of continual data server device which relatesto embodiment 3 of this invention

【図16】

同実施形態におけるマイクロスケジューラおよびマイクロスケジューラテーブルの構成の一例を示す図

[Figure 16 ]

In same embodiment , [maikurosukejuura ] and figure which shows one example of the configuration of [maikurosukejuuruteeburu ]

【図17】

同実施形態におけるマイクロスケジューラおよびマイクロスケジューラテーブルの構成の他の例を示す図

[Figure 17 ]

In same embodiment , [maikurosukejuura ] and figure which shows other example of configuration of [maikurosukejuuruteeburu ]

【図18】

同実施形態における SCSI コントローラ 14 の内部構成の一例を示す図

[Figure 18 ]

Figure which shows one example of internal configuration of SCSI controller 14 in same embodiment

【図19】

従来の連続データサーバ装置の構成の一例を示す図

[Figure 19 ]

Figure which shows one example of configuration of conventional continual data server device

【図20】

従来の連続データサーバ装置の構成の他の例を示す図

[Figure 20 ]

Figure which shows other example of configuration of conventional continual data server device

【符号の説明】

[Explanation of Symbols in Drawings ]

12

12

ディスク装置

disk drive

14

14

SCSI コントローラ

SCSI controller

16

16

ATM コントローラ

ATM controller

162

162

ネットワークプロセッサ

network processor

163

163

VBUSIF

VBUSIF

164

164

パケットメモリ

packet memory

165

165

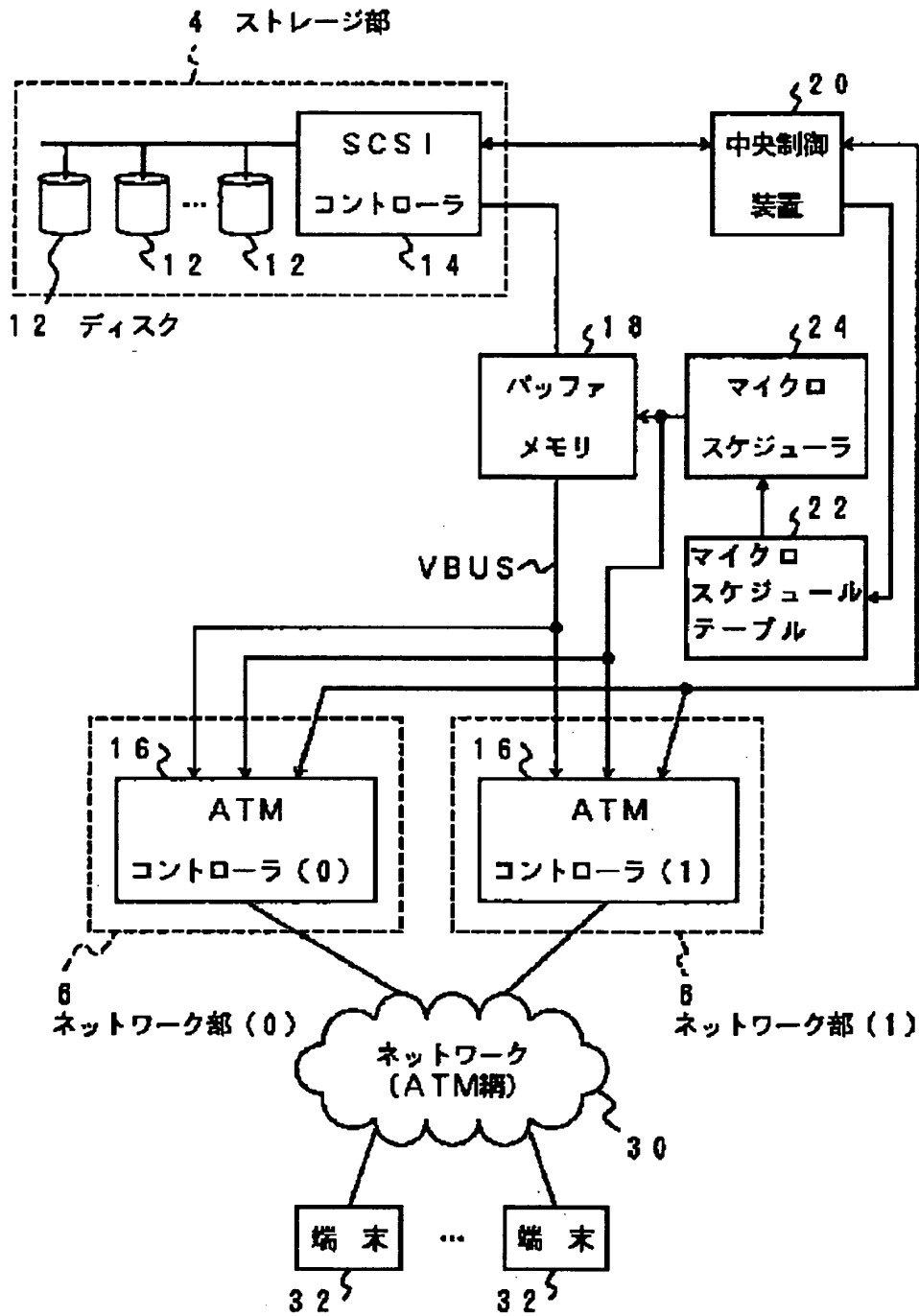
SAR	SAR
167	167
物理層インタフェース部	physical layer interface
18	18
バッファメモリ	buffer memory
185	185
SCSI	SCSI
20	20
中央制御装置	central control system
22	22
マイクロスケジュールテーブル	[maikurosukejuuruteeburu]
24	24
マイクロスケジューラ	[maikurosukejuura]
241	241
レジスタ	register
242	242
インクリメント器	increment vessel
243	243
インクリメント器	increment vessel
244	244
書き込みゲート	writing gate
245	245
マルチプレクサ	multiplexer
246	246
論理演算素子	logic computation element
247	247
論理演算素子	logic computation element
248	248
繰り返しカウンタ	Repetition counter
249	249
論理演算素子	logic computation element
250	250
繰り返しカウンタ	Repetition counter
251	251
ポインタカウンタ	pointer counter



**JP1997251437A**

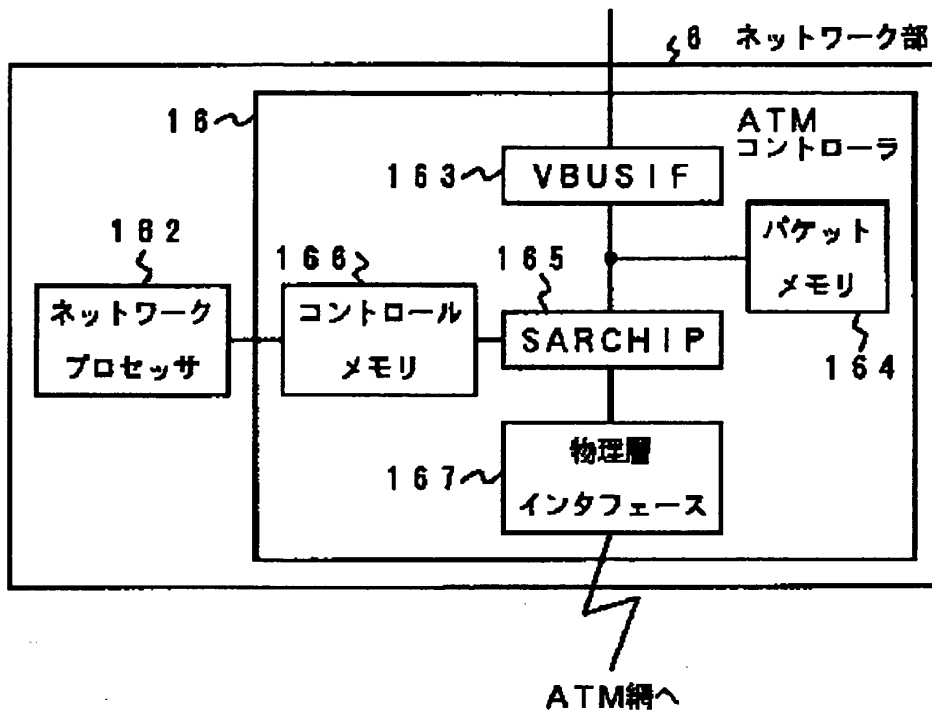
**1997-9-22**

252	252
ポインタカウンタ	pointer counter
253	253
マルチプレクサ	multiplexer
30	30
ネットワーク	network
32	32
ユーザ端末	user terminal
4	4
ストレージ部	storage unit section
6	6
ネットワーク部	network section
CHIP	CHIP
166	166
CHIP	CHIP
コントロール・メモリ	Control & memory
CHIP	CHIP
184	184
CHIP	CHIP
ストレージメモリ	storage unit memory
VBUS	VBUS
共通バス	common bus
Drawings	
【図1】	[Figure 1 ]



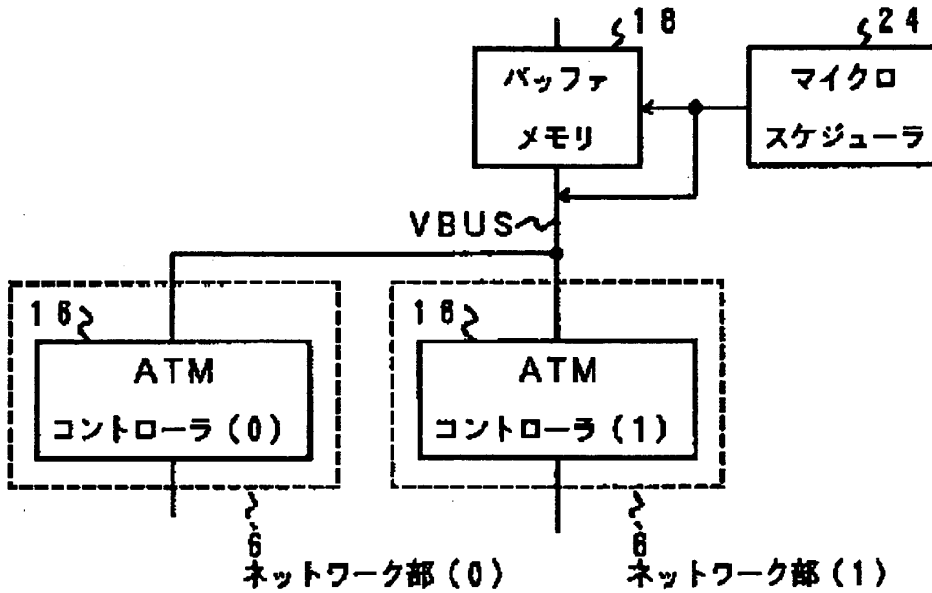
【図2】

[Figure 2]  
バッファメモリへ



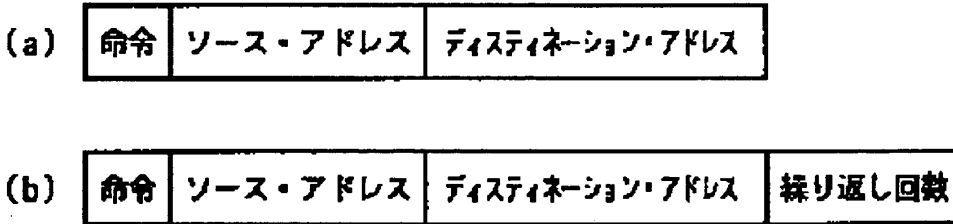
【図3】

[Figure 3]



【図4】

[Figure 4]



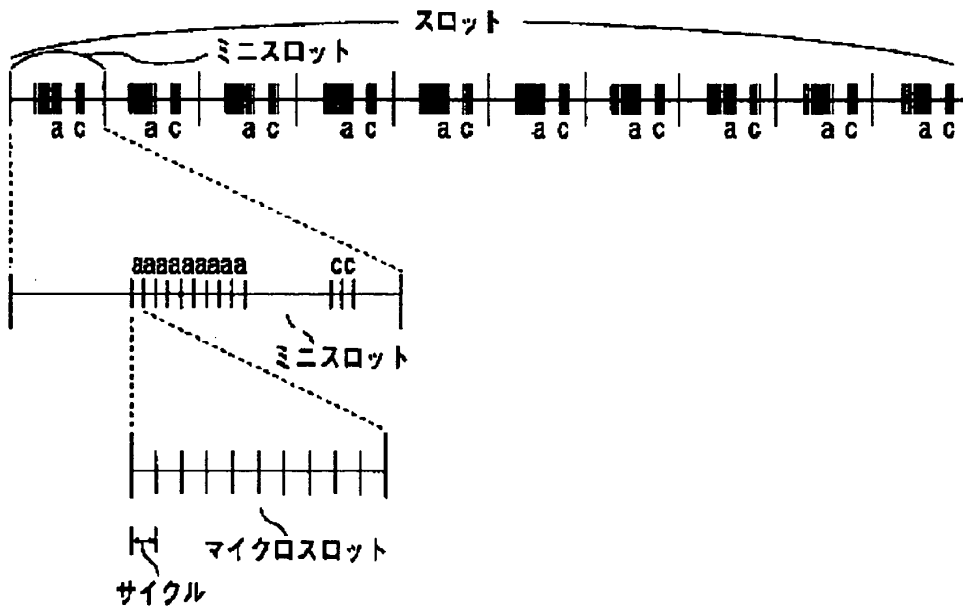
【図5】

[Figure 5]

命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ 1
命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ 2
:				:
命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ i
:				:

【図6】

[Figure 6]



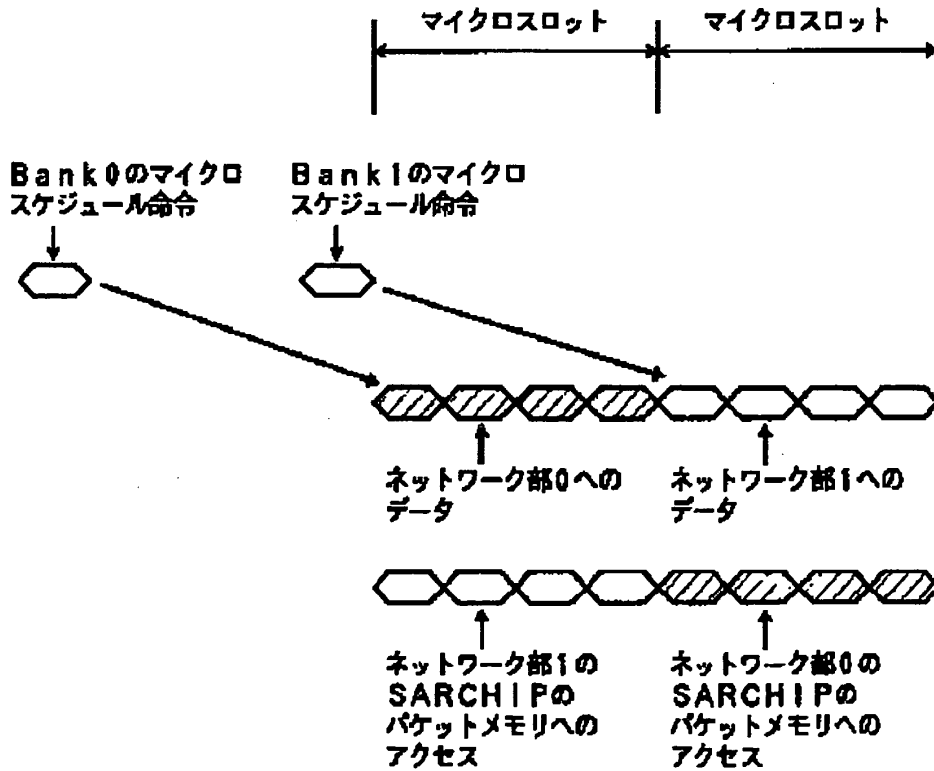
【図7】

[Figure 7]

Bank 0	命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ 1
	命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ 2
	:				:
Bank 1	命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	
	:				

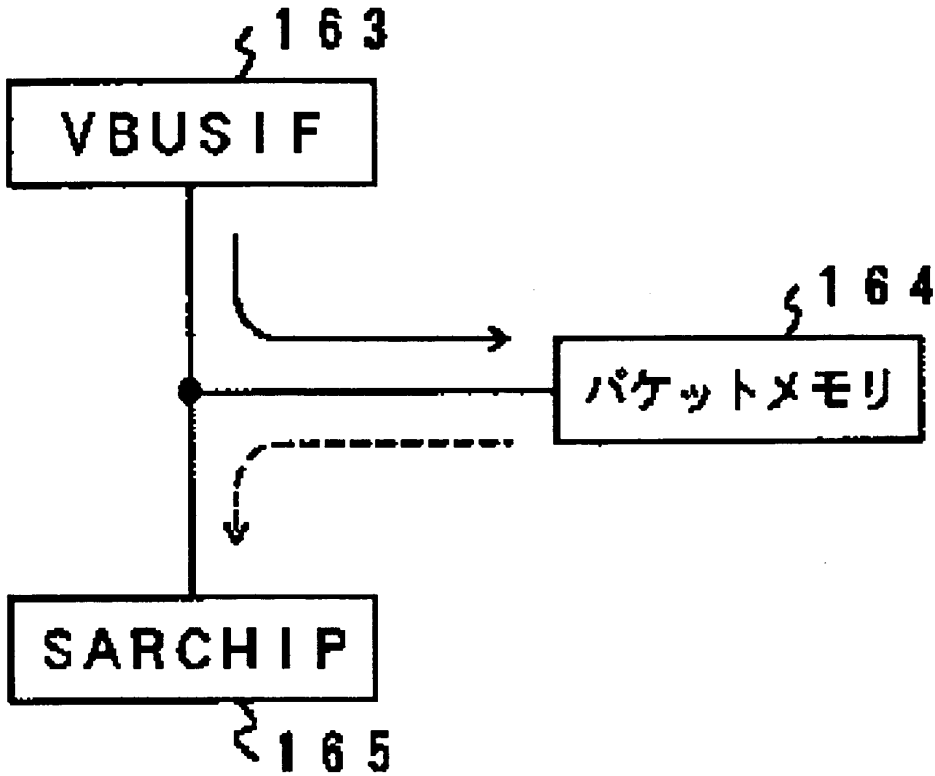
【図8】

[Figure 8]



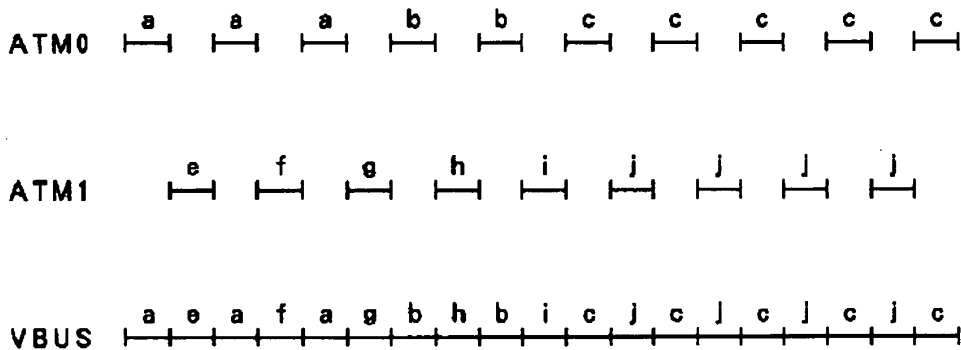
【図9】

[Figure 9]



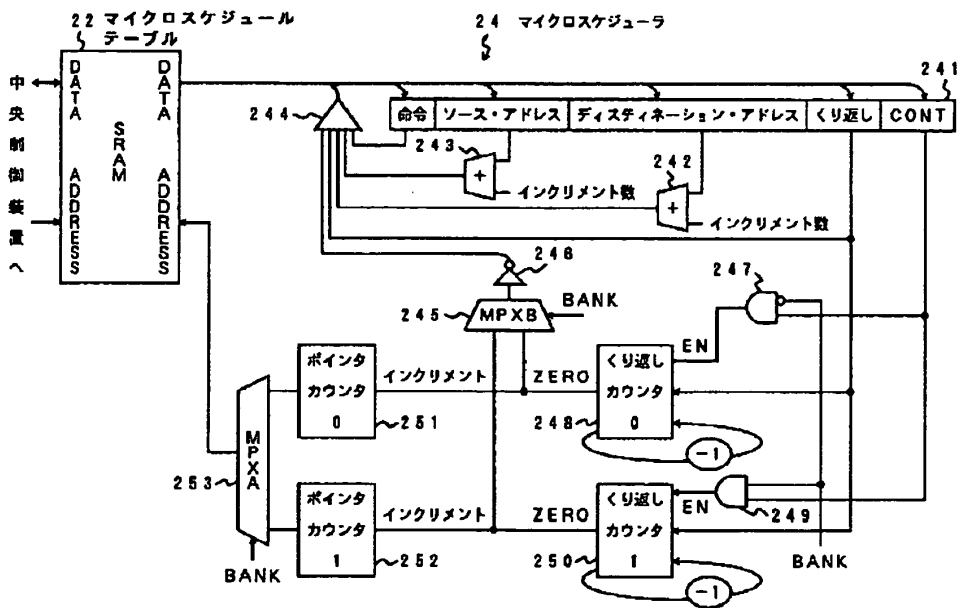
【図10】

[Figure 10]



【図11】

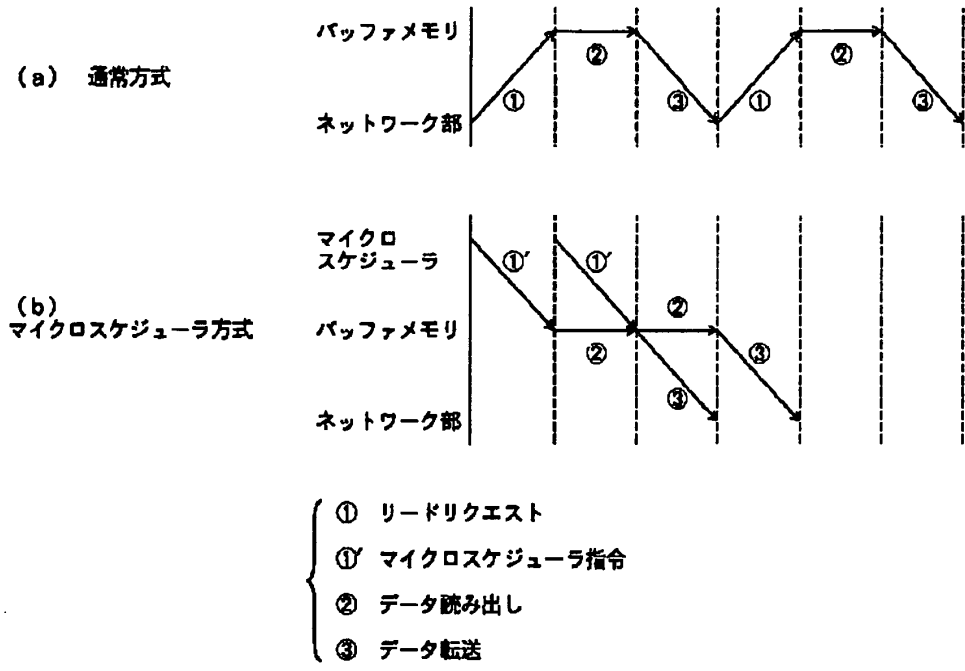
[Figure 11]



【図12】

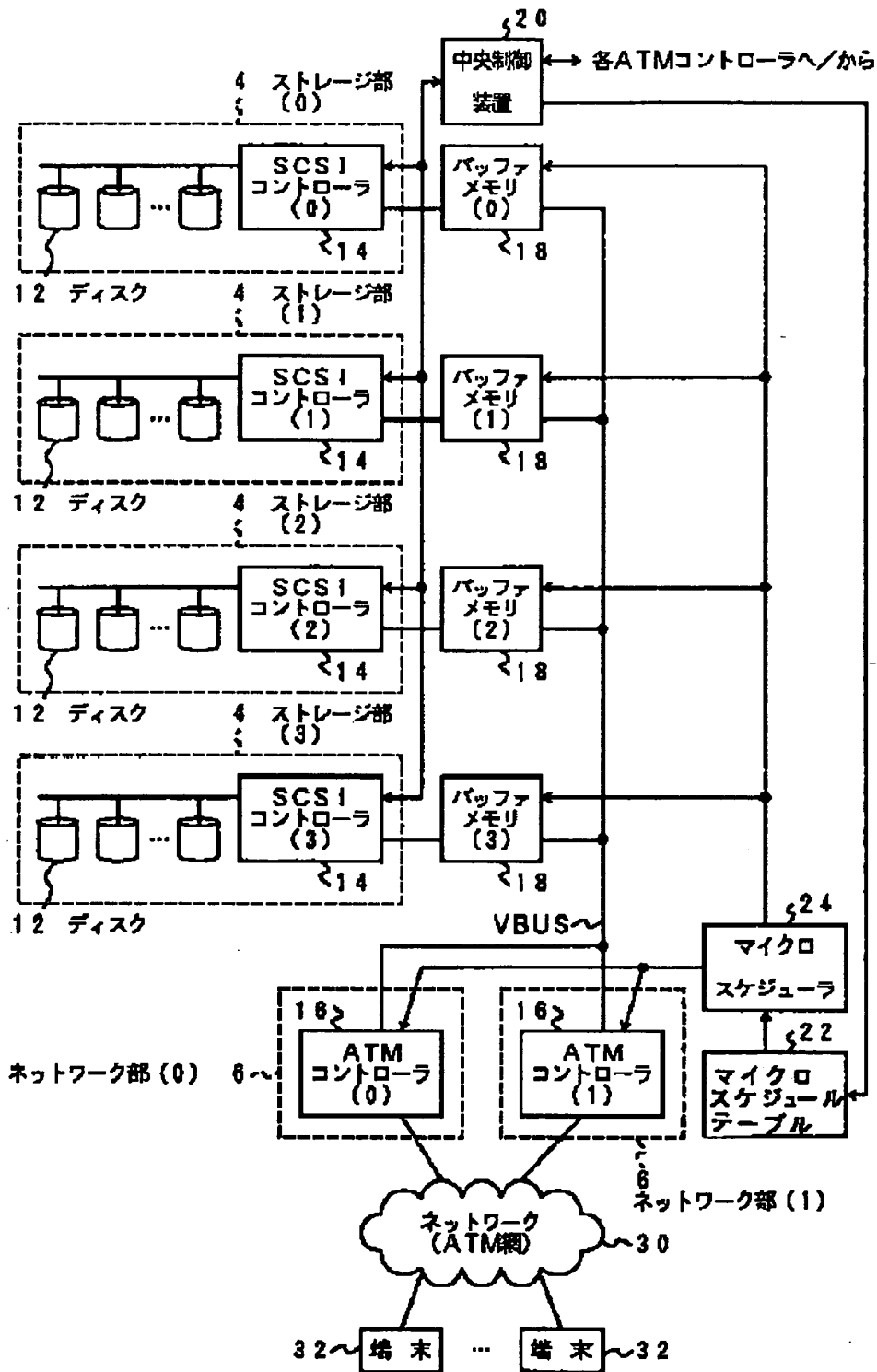
[Figure 12]





【図13】

[Figure 13]

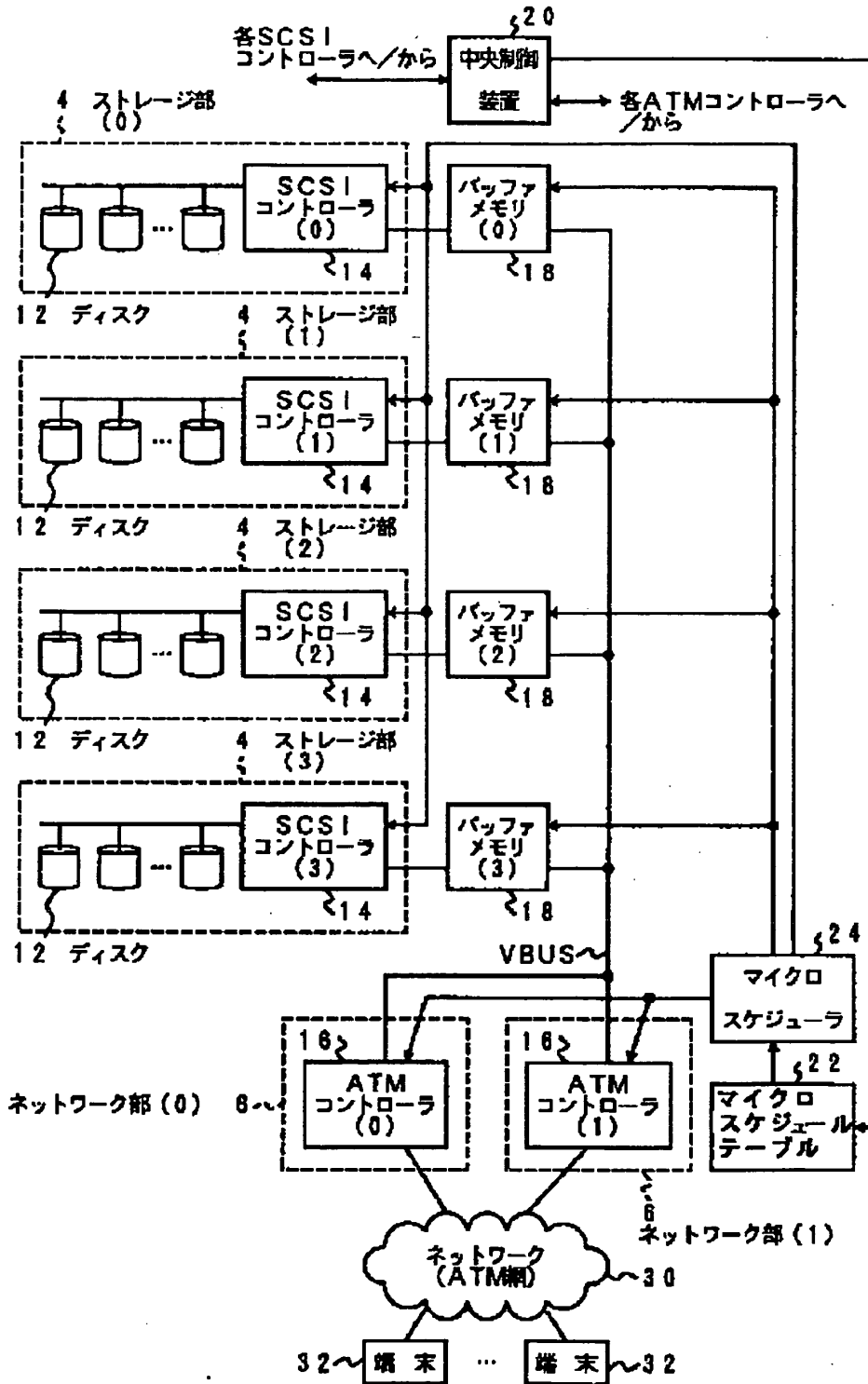


**JP1997251437A**

**1997-9-22**

**【図15】**

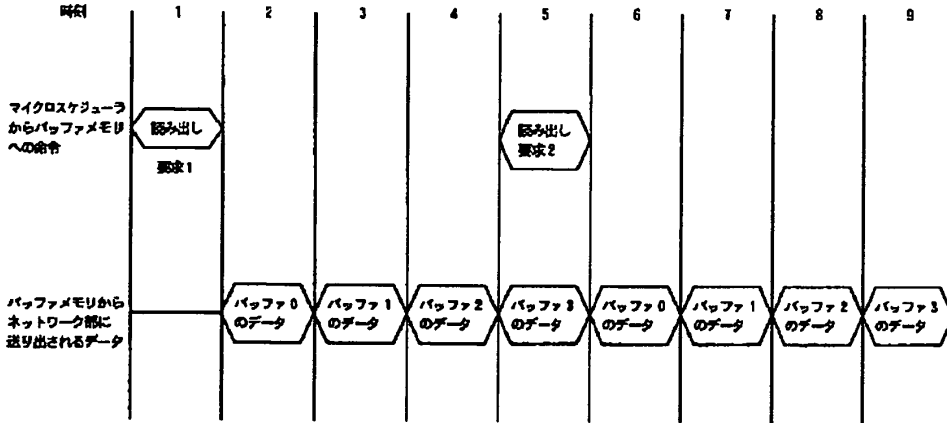
**[Figure 15]**



(7,296)

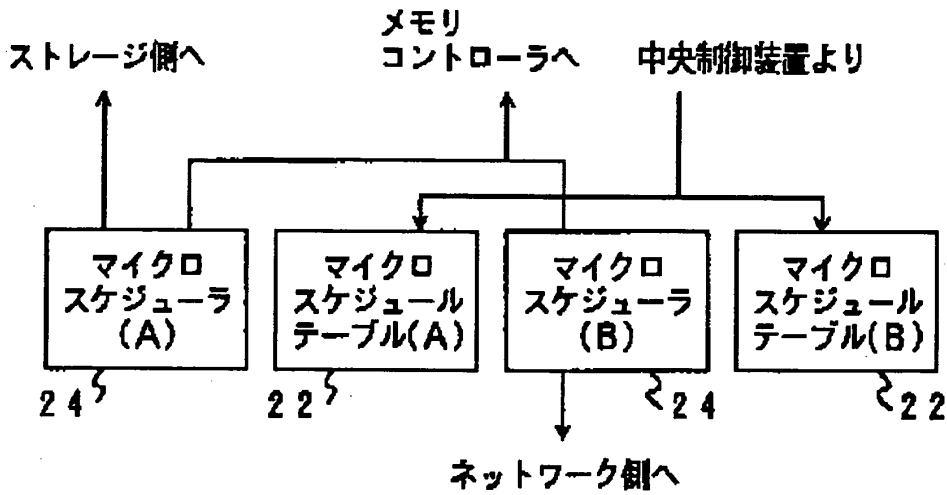
【図14】

[Figure 14]



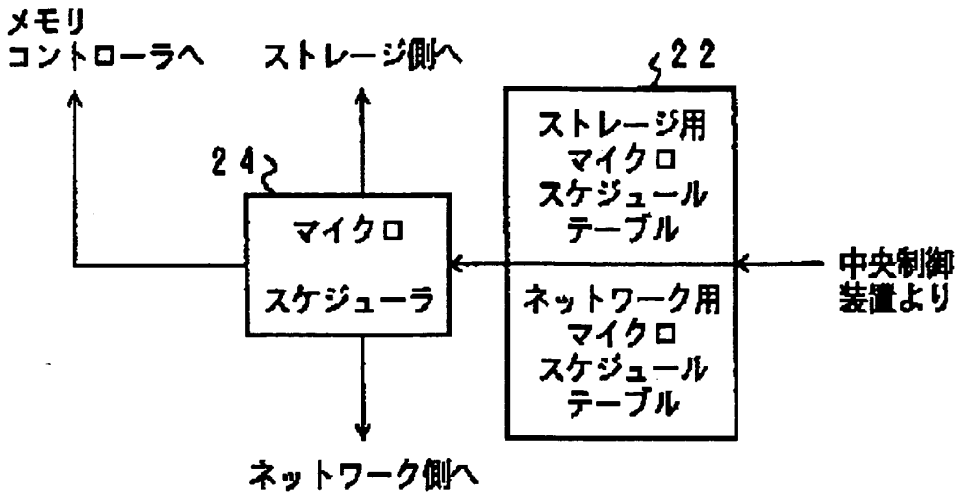
【図16】

[Figure 16]



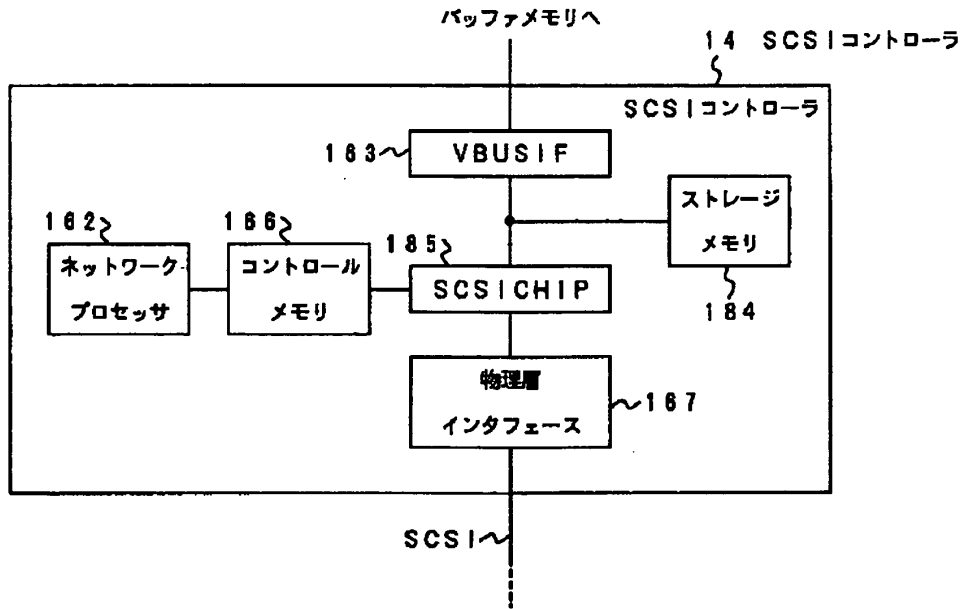
【図17】

[Figure 17]



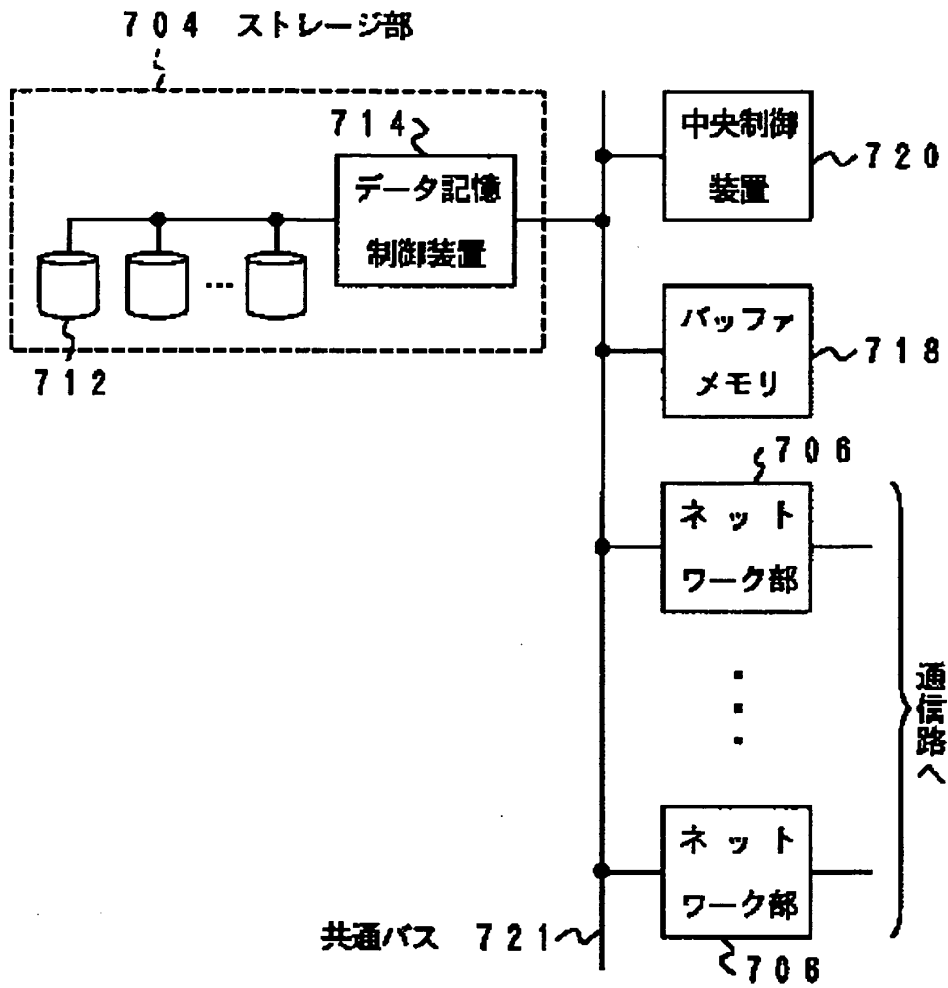
【図18】

[Figure 18]



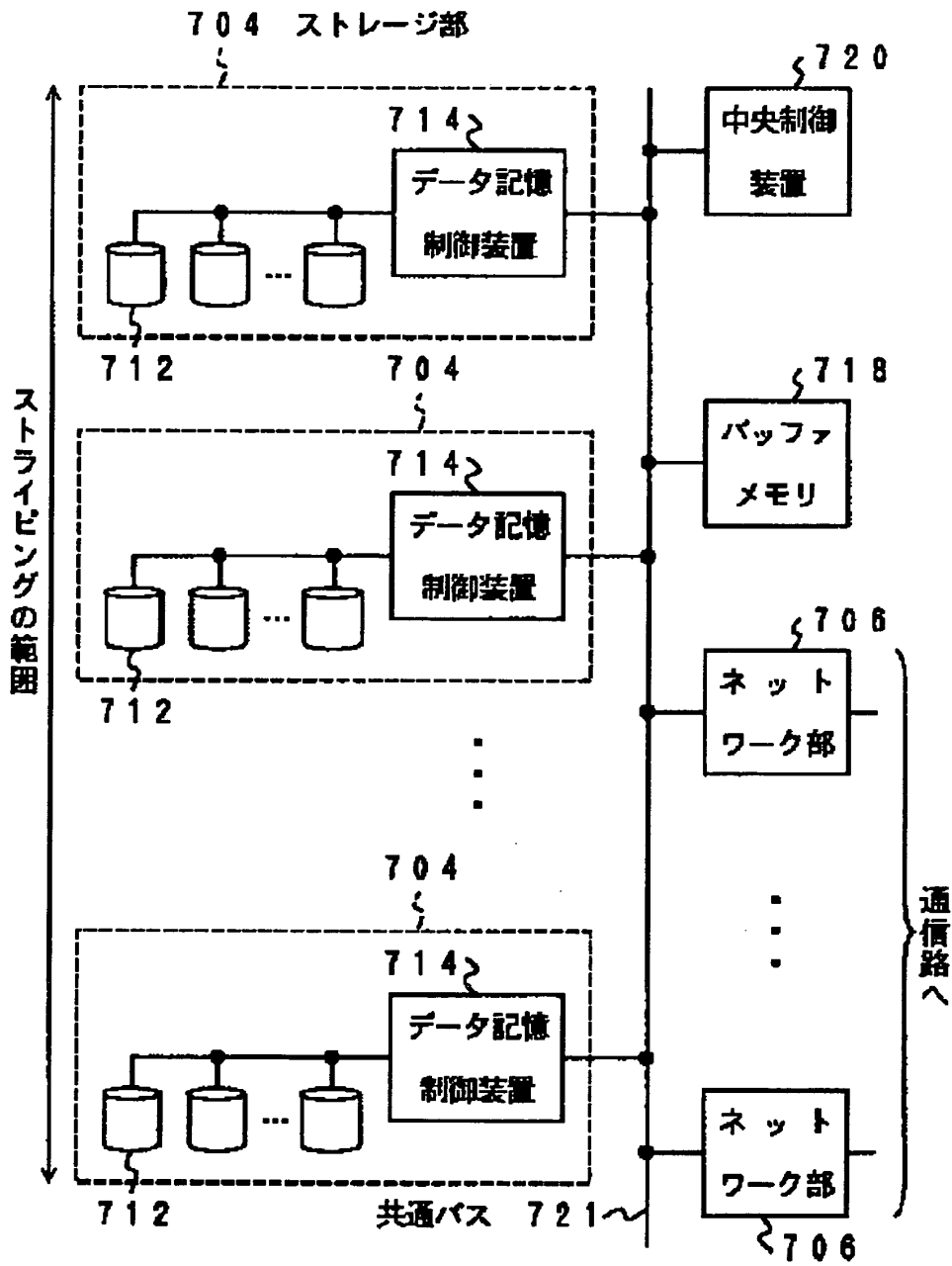
【図19】

[Figure 19]



【図20】

[Figure 20]





PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09251437 A**

(43) Date of publication of application: **22.09.97**

(51) Int. Cl. **G06F 13/372**  
**G06F 13/00**  
**G06F 13/16**

(21) Application number: **08061467**

(71) Applicant: **TOSHIBA CORP**

(22) Date of filing: **18.03.98**

(72) Inventor: **ASANO SHIGEHIRO**  
**SUZUKI MAKI**

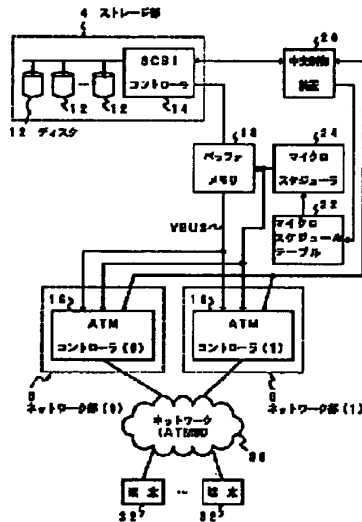
(54) **COMPUTER DEVICE AND CONTINUOUS DATA SERVER DEVICE**

(57) Abstract:

**PROBLEM TO BE SOLVED:** To improve the use rate of a bus by permitting an allocation means to allocate a right for using the common bus by means of respective units in terms of decision theory.

**SOLUTION:** A central controller 20 issues operation commands in a slot (prescribed time interval) unit to a storage part 4 and a network part 6 and Writes the program of a micro schedule into a micro schedule table 22. A micro scheduler 24 definitely controls the use of the common bus YBUS by the storage part 4 and the network part 6 in accordance with the program. Namely, the bus use periods of the respective units are definitely guaranteed not by mediating a bus use right from the respective units but allocating the bus use right by the previously decided micro schedule.

COPYRIGHT: (C)1997,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-251437

(43) 公開日 平成9年(1997)9月22日

(51) Int.Cl. <sup>8</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 13/372			G 0 6 F 13/372	C
	13/00	3 5 3	13/00	3 5 3 N
	13/16	5 2 0	13/16	5 2 0 B

審査請求 未請求 請求項の数15 O L (全 18 頁)

(21) 出願番号 特願平8-61467

(22) 出願日 平成8年(1996)3月18日

(71) 出願人 00003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 浅野 滋博

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝研究開発センター内

(72) 発明者 鈴木 真樹

神奈川県川崎市幸区小向東芝町1番地 株

式会社東芝研究開発センター内

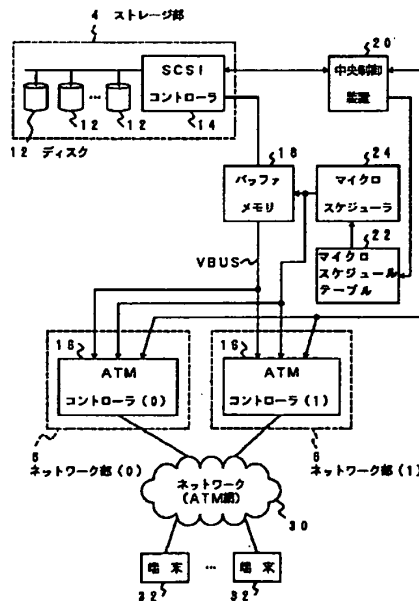
(74) 代理人 弁理士 鈴江 武彦

(54) 【発明の名称】 計算機装置及び連続データサーバ装置

(57) 【要約】

【課題】 共通バスの使用率の向上を可能とする計算機装置を提供すること。

【解決手段】 複数のユニット6と、複数のユニットを接続する共通バスVBUSと、プログラムを格納する格納手段22と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる割当手段24とを備えたことを特徴とする。好ましくは、プログラムを作成し、前記格納手段に書き込む手段20をさらに備える。また、好ましくは、前記格納手段は複数のメモリバンクからなり、あるバンクへの中央制御装置からの書き込みと、他のバンクからの前記割当手段による読み出しとが並行的に行われる。また、好ましくは、前記割当手段は、前記プログラムに従って、各ユニットにアドレス及び動作モードを転送する。



## 【特許請求の範囲】

【請求項1】複数のユニットと、  
複数のユニットを接続する共通バスと、  
プログラムを格納する格納手段と、  
このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる割当手段とを備えたことを特徴とする計算機装置。

【請求項2】前記プログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする請求項1に記載の計算機装置。

【請求項3】前記格納手段は複数のメモリバンクからなり、

あるバンクへの中央制御装置からの書き込みと、他のバンクからの前記割当手段による読み出しとが並列に行われることを特徴とする請求項2に記載の計算機装置。

【請求項4】前記割当手段は、前記プログラムに従って、各ユニットにアドレス及び動作モードを転送するものであることを特徴とする請求項1に記載の計算機装置。

【請求項5】前記格納手段に格納されるプログラムは、読み出し先アドレス、書き込み先アドレス及び繰り返し回数を含むエントリからなり、

前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送して該アドレスをインクリメントする動作を、前記繰り返し回数に従って繰り返すものであることを特徴とする請求項1に記載の計算機装置。

【請求項6】前記割当手段は、書き込み先ユニットに書き込み先アドレスを含む指示を転送した場合に、この指示に対応するデータが共通バス上に存在しなかったならば、  
該アドレスのインクリメントを行わず、書き込み先ユニットに対し書き込みを行わないよう指示することを特徴とする請求項5に記載の計算機装置。

【請求項7】前記格納手段は複数のメモリバンクからなり、

各メモリバンクにはそれぞれ対応するターゲットユニットへの指示の基となるプログラムが格納されるものであることを特徴とする請求項1に記載の計算機装置。

【請求項8】前記割当手段は前記ターゲットユニットのそれぞれが前記共通バスをインタリーブして使用するよう割り当てることを特徴とする請求項7に記載の計算機装置。

【請求項9】ストレージから読み出したデータを一時記憶するバッファメモリユニットと、  
バッファメモリのデータを各ユーザを宛先として通信路に送り出す通信制御ユニットと、  
これらユニットを接続する共通バスと、  
プログラムを格納する手段と、  
このプログラムに従って、各ユニットが共通バスを使用

する権利を、決定論的に割り当てる手段とを備えたことを特徴とする連続データサーバ装置。

【請求項10】各ユーザ宛に送り出されるべきデータが一定の間隔でバッファメモリユニットから通信制御ユニットに取り込まれるように、共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする請求項9に記載の連続データサーバ装置。

【請求項11】通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、

前記スロットを複数に等分に分割した各ミニスロット内の一定の位置に、各ユーザ宛に送り出されるべきデータを通信制御ユニットに取り込むための共通バスの使用が割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする請求項9に記載の連続データサーバ装置。

【請求項12】前記中央制御装置が書き込むプログラムは、前記ミニスロット単位で作成され、  
前記割当手段は、スロットの分割数分のこのプログラムを繰り返して用いるものであることを特徴とする請求項11に記載の連続データサーバ装置。

【請求項13】通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、

各ユーザ宛に送り出されるべきデータのビットレートに応じた数の、前記スロットを複数に等分に分割したマイクロスロットが、該データを通信制御ユニットに取り込むための共通バスの使用期間として割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする請求項9に記載の連続データサーバ装置。

【請求項14】前記中央制御装置が書き込むプログラムは、バッファメモリユニット及び通信制御ユニット内のアドレス、前記マイクロスロットの数に対応する繰り返し回数を含むエントリからなり、

前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送する動作を、前記繰り返し回数に従って繰り返すことにより、前記ビットレートに応じた使用期間を割り当てるものであることを特徴とする請求項13に記載の連続データサーバ装置。

【請求項15】ストレージから読み出されたデータをバッファメモリユニットに取り込む際の共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする請求項9に記載の連続データサーバ装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、共通バスに接続さ

れた複数のユニットを持つ計算機装置および、少なくとも一部のユニット間のデータ転送に共通バスを利用する連続データサーバ装置に関する。

【0002】

【従来の技術】デジタル情報を扱う計算機システムで複数のユニットを接続する安価な方法として共通バスが広く採用されている。共通バスを使用したシステムを構築する上での問題点は共通バスのバンド幅ネックが発生し易いことである。共通バスのバンド幅を増やすためには信号線の数を増やすか、共通バスの動作周波数を上げることが考えられるがいずれの場合にもコストの増大を招くことになる。

【0003】コストの増大を避けながら共通バスのバンド幅ネックを防ぐ効果的な方法としては共通バスの使用率の向上が考えられる。しかし一般に共通バスに複数のユニットを接続した場合、共通バスの使用権をめぐる調停動作が必要であり、調停動作には一定の時間が必要なので、これが共通バスの使用率を高める上で障害となっていた。

【0004】また、メモリを多数用いる計算機システムではメモリとして安価なDRAMが使用されることが多い。DRAMは高速ページモードなどで使用するとバンド幅は大きくとれるが、レーテンシも大きいという問題がある。従って、DRAMと共通バスの間でデータ転送が行われる場合、調停動作にレーテンシが加わり、さらに共通バスの使用率を低下させる。

【0005】ところで、共通バスを用いる計算機システムの1つに、連続データサーバ装置がある。映像や音声のような連続データを扱う連続データサーバ装置は、記憶装置に記憶した連続データを読み出し、端末装置に対して時間に同期しながらリアルタイムに連続的に(一定期間内に一定量の)データを送り出す機能を持つ。このような連続データサーバ装置は、複数の映画等のビデオデータを記憶し端末からの要求に応じて任意の映画を送り出すビデオ・オン・デマンド、ネットワークを介して映像による商品情報を提供するオンラインショッピングなどの分野で使用される。それゆえ、連続データサーバ装置には、複数のユーザからランダムに要求が送られてくるので、同時に多数のユーザに対してそれぞれ異なる連続データの送り出しを間断なく行なう能力が要求されることになる。

【0006】以下、従来の連続データサーバ装置について詳しく説明する。従来の連続データサーバ装置の1つに、図19に示すような構成を持つものがある。このような連続データサーバ装置において、ユーザあるいはアプリケーションプログラムによって発せられた連続データへのアクセス要求が、プロセス間通信やネットワークを経由した通信等によって送られてくる。このアクセス要求は、ネットワーク部706から共通バス721を介して中央制御装置720に伝えられ、受理される。中央

制御装置720は、要求された連続データの読み出しをストレージ部704に伝える。ストレージ部704のデータ記憶制御装置714は、指示された連続データをデータ記憶装置712から読み出して、バッファメモリ718に書き込む。中央制御装置720は、バッファメモリ718上のデータの送り出しをネットワーク部706に指示する。ネットワーク部706は、アクセス要求にて指定された転送先に対し、連続データを送り出す。これら動作は、通常、スロットと呼ばれる一定の時間間隔を単位として行われる。連続データを記憶するデータ記憶装置には、ディスク装置を用いる場合もあるが、光ディスクや光磁気ディスク装置等を用いる場合もある。ディスク装置以外にも、RAMやEEPROM等の半導体記憶装置を用いることもある。

【0007】また、図20に示すように複数系列のストレージ部704を備え、1つの連続データを分割して複数系列に渡って分散格納するストライピング技法を適用したものがあ。これは、データ転送能力(総バンド幅)を大きくし、同一の連続データへより多数のユーザが同時にアクセスすることを可能にすることを目的としている。

【0008】この種の従来の連続データサーバ装置では、ストレージ部704やネットワーク部706が共通バス721の使用要求を出し、図示しない調停装置が調停を行い、使用権を獲得したストレージ部704やネットワーク部706が共通バス721を使用してバッファメモリ718との間のデータ転送を行う。従って、調停動作に要する時間が共通バス721の使用率を低下させ、同時にアクセスできるユーザ数が低下されてしまう。また、連続データサーバ装置ではユーザ端末側の再生を中断させないために、各ユーザ端末に一定の時間間隔で一定量のデータを間断なく転送することを常に保証する必要があるため、この保証のために、バスの使用率を余裕をとって低く抑えておく。従って、上記保証をしつつ、共通バスの使用率を向上させる新たな技術が望まれている。

【0009】

【発明が解決しようとする課題】従来、共通バスに接続された複数のユニットを持つ計算機装置では、共通バスの使用権をめぐる調停動作に一定の時間が浪費され、これが共通バスの使用率を高める上で障害となっていた。また、DRAMと共通バスの間でデータ転送が行われる場合、調停動作にレーテンシが加わり、さらに共通バスの使用率を低下させる問題があった。

【0010】また、従来、装置内部のユニット間データ転送に共通バスを利用する連続データサーバ装置では、あるユーザ端末について一定の時間間隔で一定量のデータを間断なく転送することを常に保証しつつ、共通バスの使用率を向上させることは困難であった。

【0011】本発明は、上記事情に鑑みてなされたもの

であり、共通バスの使用率の向上を可能とする計算機装置を提供することとする。また、本発明は、各ユーザ端末に一定の時間間隔で一定量のデータを間断なく転送することを常に保証しつつ、装置内部でのデータ転送に用いる共通バスの使用率の向上を可能とする連続データサーバ装置を提供することとする。

【0012】

【課題を解決するための手段】本発明（請求項1）に係る計算機装置は、複数のユニット（下記マイクロスケジューラの割り当てに従って共通バスを使用する）と、複数のユニットを接続する共通バスと、プログラムを格納する格納手段（マイクロスケジューラテーブル）と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる割当手段（マイクロスケジューラ）とを備えたことを特徴とする。

【0013】各ユニットからのバス使用要求に依りてバス使用権の調停を行うのではなく、予め定めたスケジューリングによりバス使用権を割り当てるため、各ユニットのバス使用時期を確定的に保証することができる。さらに、これを利用して（メモリへの指示を先行発行することにより）、メモリのレーテンシを隠蔽することができる。従って、バスの効率的な使用（使用率の向上）が実現できる。

【0014】本発明（請求項2）は、請求項1において、前記プログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする。本発明（請求項3）は、請求項2において、前記格納手段は複数のメモリバンクからなり、あるバンクへの中央制御装置からの書き込みと、他のバンクからの前記割当手段による読み出しとが並列的に行われることを特徴とする。

【0015】本発明（請求項4）は、請求項1において、前記割当手段は、前記プログラムに従って、各ユニットにアドレス（読み出し先アドレス／書き込み先アドレス）及び動作モード（読み出し命令／書き込み命令）を転送するものであることを特徴とする請求項1に記載の計算機装置。

【0016】本発明（請求項5）は、請求項1において、前記格納手段に格納されるプログラムは、読み出し先アドレス、書き込み先アドレス及び繰り返し回数を含むエントリからなり、前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送して該アドレスをインクリメントする動作を、前記繰り返し回数に従って繰り返すものであることを特徴とする。本発明（請求項6）は、請求項5において、前記割当手段は、書き込み先ユニットに書き込み先アドレスを含む指示を転送した場合に、この指示に対応するデータが共通バス上に存在しなかったならば、該アドレスのインクリメントを行わず、書き込み先ユニットに対し書き込みを行わないよう指示することを特徴とする。

【0017】本発明（請求項7）は、請求項1において、前記格納手段は複数のメモリバンクからなり、各メモリバンクにはそれぞれ対応するターゲットユニットへの指示の基となるプログラムが格納されるものであることを特徴とする。

【0018】本発明（請求項8）は、請求項7において、前記割当手段は前記ターゲットユニットのそれぞれが前記共通バスをインタリーブして使用するよう割り当てることを特徴とする。

10 【0019】本発明（請求項9）に係る連続データサーバは、ストレージから読み出したデータを一時記憶するバッファメモリユニットと、バッファメモリのデータを各ユーザを宛先として通信路に送り出す通信制御ユニット（これらユニットは下記のマイクロスケジューラの割り当てに従って共通バスを使用する）と、これらユニットを接続する共通バスと、プログラムを格納する手段と、このプログラムに従って、各ユニットが共通バスを使用する権利を、決定論的に割り当てる手段とを備えたことを特徴とする。

20 【0020】請求項1の発明について説明した作用効果に加え、一定の間隔で通信制御ユニットに連続データが送り込まれ、ひいては、一定の間隔で通信路に連続データが送り出されることを保証することができる。従って、通信制御ユニットが通信路に連続データを送り出すまで該データを一時記憶するバッファメモリの容量を少なくでき、また、通信路の先に存在して連続データを再生するユーザ端末において受信したバッファを保持するバッファの容量を少なくできる。

30 【0021】本発明（請求項10）は、請求項9において、各ユーザ宛に送り出されるべきデータが一定の間隔でバッファメモリユニットから通信制御ユニットに取り込まれるように、共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする。

【0022】本発明（請求項11）は、請求項9において、通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、前記スロットを複数に等分に分割した各ミニスロット内の一定の位置に、各ユーザ宛に送り出されるべきデータを通信制御ユニットに取り込むための共通バスの使用が割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする。

【0023】これにより、スロットよりも細かい単位の一一定の間隔で、通信制御ユニットに連続データが送り込まれることを保証することができる。従って、バッファメモリやユーザ端末のバッファの容量をさらに少なくできる。

50 【0024】本発明（請求項12）は、請求項11において、前記中央制御装置が書き込むプログラムは、前記

ミニスロット単位で作成され、前記割当手段は、スロットの分割数分のこのプログラムを繰り返して用いるものであることを特徴とする。

【0025】これにより、前記格納手段の容量を小さくできる。本発明（請求項13）は、請求項9において、通信制御ユニットを介したバッファメモリから通信路へのデータの送り出しのタイミングをスロット単位で指示する手段と、各ユーザ宛に送り出されるべきデータのビットレートに応じた数の、前記スロットを複数に等分に分割したマイクロスロットが、該データを通信制御ユニットに取り込むための共通バスの使用期間として割り当てられるように、プログラムを作成し、前記格納手段に書き込む手段とを有する中央制御装置をさらに備えたことを特徴とする。

【0026】これにより、各ユーザの要求ビットレートに応じたバス使用期間を、各スロット内で確保することができる。クレーム11のミニスロットと組み合わせて（ミニスロットをさらに分割したものをマイクロスロットとして）用いるとさらに良い。

【0027】本発明（請求項14）は、請求項13において、前記中央制御装置が書き込むプログラムは、バッファメモリユニット及び通信制御ユニット内のアドレス、前記マイクロスロットの数に対応する繰り返し回数を含むエントリからなり、前記割当手段は、このエントリに含まれるアドレスに基づいて各ユニットへの指示を転送する動作を、前記繰り返し回数に従って繰り返すことにより、前記ビットレートに応じた使用期間を割り当てるものであることを特徴とする。

【0028】これにより、前記格納手段の容量を小さくできる。本発明（請求項15）は、請求項9において、ストレージから読み出されたデータをバッファメモリユニットに取り込む際の共通バスの使用を割り当てるプログラムを作成し、前記格納手段に書き込む手段を有する中央制御装置をさらに備えたことを特徴とする。また、請求項10の中央制御装置に本手段を備えて構成することもできる。

【0029】なお、請求項2～8の各発明は、請求項9の連続データサーバ装置にも適用可能である。請求項6を請求項9に適用する場合、前記割当手段は、バッファメモリユニットに書き込み先アドレスを含む指示を転送した場合に、この指示に対応するデータが共通バス上に存在しなかったならば、該アドレスのインクリメントを行わず、バッファメモリユニットに対し書き込みを行わないよう指示するようにする。

【0030】請求項7を請求項9に適用する場合、前記格納手段は複数の通信制御ユニット対応に設けられたメモリバンクからなり、各メモリバンクにはそれぞれ対応する通信制御ユニットへの指示の基となるプログラムを格納されるようにする。

【0031】請求項8を請求項9に適用する場合、前記

割当手段は前記通信制御ユニットへのそれぞれに対する指示をインタリーブして転送するようにする。また、請求項11～14の各発明は、請求項15の連続データサーバ装置にも適用可能である。これらの場合、請求項15の中央制御装置は、ストレージからバッファメモリユニットへのデータの読み出しのタイミングをスロット単位で指示する手段をさらに含み、前記共通バスの使用の割り当ては、前記スロットを複数に等分に分割したマイクロスロット単位で行っても良い。

10 【0032】なお、前記格納手段はメモリを用いて構成しても良い。また、前記格納手段に格納するプログラムとして何もしないNOP命令を格納できるようにしても良い。

【0033】また、各ユニットにアドレス及び動作モードを転送するために、データを転送する転送路（共通バス）と同じ転送路を使用しても良いし、該データを転送する転送路とは別に設けた転送路を使用しても良い。

【0034】

20 【発明の実施の形態】以下、図面を参照しながら発明の実施の形態を説明する。本実施形態では、本発明に係るマイクロスケジューラによる決定論的な割り当てに従って共通バスを使用する複数のユニットを有する計算機装置として、複数の連続データのアクセス要求に同時に応答して映像や音声等の連続データのサービスを行う連続データサーバ装置を取り上げる。

【0035】第1の実施形態および第2の実施形態では本発明を連続データサーバのネットワーク部に適用した場合について、第3の実施形態では本発明を連続データサーバ装置のネットワーク部およびストレージ部に適用した場合について説明する。

30 【0036】（実施形態1）最初に、実施形態1について説明する。図1に本実施形態に係る連続データサーバ装置の構成を示す。

40 【0037】本実施形態の連続データサーバ装置は、連続データを格納している所定台数のデータ記憶装置12と該データ記憶装置12からデータを読み出すデータ記憶制御装置14とからなるストレージ部4、該ストレージ部4から読み出したデータを一時記憶するバッファメモリ18、該バッファメモリ18のデータを各ユーザ端末32を宛先としてネットワーク30に送り出す複数（ここでは2台とする）のネットワーク部6、ストレージ部4とバッファメモリ18とネットワーク部6を接続する共通バスVBUS、端末32からの要求に基づいてデータ記憶装置12からデータを読み出してネットワーク30に送り出すまでのスケジューリング、ネットワークの設定をはじめとして、システム全体の制御を行なう中央制御装置20、該中央制御装置20により確定されたマイクロスケジューラのプログラムを格納するマイクロスケジューラテーブル22、該プログラムに従ってネットワーク部6に共通バスVBUSを使用する権利を確定的

に割り当てるマイクロスケジューラ24を備えている。

【0038】ストレージ部4、ネットワーク部6、バッファメモリ18の役割は夫々図19や図20の対応するユニットと基本的には同様であるが、本実施形態では、中央制御装置20はストレージ部4とネットワーク部6にスロット(一定の時間間隔)単位の動作指令を出すとともに、マイクロスケジューラ22にマイクロスケジューラのプログラムを書き込み、マイクロスケジューラ22はこのプログラムに従いストレージ部4およびネットワーク部6による共通バスVBUSの使用を確定的に制御していく。すなわち、各ユニットからのバス使用要求に依存してバス使用権の調停を行うのではなく、予め定めたマイクロスケジューラによりバス使用権を割り当てることにより、各ユニットのバス使用時期を確定的に保証するようにしている。これによって、一定の間隔でネットワーク部6に一定量の連続データが送り込まれ、一定の間隔で通信路に一定量の連続データが送り出されることを保証しつつ、共通バスの使用率の向上を図ることができ、同時サービス可能なユーザ数を向上させることができる。また、ネットワーク部6が通信路に連続データを送り出すまで該データを一時記憶するバケットメモリ18の容量を少なくでき、また、通信路の先に存在して連続データを再生するユーザ端末において受信したバケットを保持するバッファの容量を少なくできる利点もある。

【0039】以下、本実施形態をさらに詳しく説明していく。まず、中央制御装置20、ストレージ部4、ネットワーク部6の基本的な構成について説明する。

【0040】中央制御装置20は、例えば電子計算機と同じようにCPUとメモリ装置から構成し、システム全体に対する制御を記述したソフトウェアをCPUで実行することにより、その機能を得ることができる。中央制御装置20は、システム全体を制御するために、システム内に記憶している各連続データの仕様、各連続データのディスク装置12への配置状態、各ネットワーク部6が接続できる通信路など、システム内の情報をすべて管理している、あるいは知ることができる。連続データの仕様としては、連続データ名あるいはIDコードなどの各連続データを特定するための情報の他に、例えば各連続データの全データ長や、連続データが複数のブロックからなる場合の全ブロック数などが考えられる。

【0041】データ記憶装置12は、映像や音声等の連続データを記憶するためのものであり、磁気ディスク装置、光ディスク装置、光磁気ディスク装置等のディスク装置を用いることができる。また、ディスク装置以外にも、RAMやEEPROM等の半導体記憶装置など種々のものを用いることができる。データ記憶装置12に記憶する連続データは、連続したビットあるいはバイトの並んだ構造を持つデータである。連続データは、好ましくは、ブロックなど纏まりの単位で記憶・管理され

る。また、データ記憶制御装置14は夫々、データ記憶装置12に記憶された連続データを読み出し、これをバッファメモリ18の指示されたアドレスに書き込む。本実施形態では、データ記憶装置12としてSCSIインタフェースを持つ磁気ディスク装置を用いるものとし、以下、データ記憶装置12をディスク12、データ記憶制御装置14をSCSIコントローラ14と呼んで、説明を行うものとする。なお、連続データの代表例はビデオ・データであり、この場合連続データサーバ装置は一般的にビデオサーバと呼ばれる。

【0042】ネットワーク部6は、バッファメモリ18の指定されたアドレスから連続データを読み出し、これをネットワーク30の通信路に対して送り出す。ネットワーク部6には、ATM網やイーサネット、FDDIなどを接続することができる。本実施形態では、ネットワーク30をATM網とし、ネットワーク部6はATMコントローラ16を用いて構成するものとして説明するが、他のプロトコルに基づくネットワークに接続する場合は、接続するネットワークに応じたインタフェース機能をネットワーク部6内に適宜設ければ良い。

【0043】ここで、本実施形態のネットワーク部6の構造について説明する。図2にネットワーク部6の内部構成の一例を示す。図のようにネットワーク部6はATMコントローラ16およびネットワークプロセッサ162から構成される。

【0044】ネットワークプロセッサ162は、中央制御装置20の指示に基づいて、ATMコントローラ16のコントロール・メモリ(Control Memory)166の設定を行なう。コントロール・メモリ166には、ATMの回線の設定、バケットメモリ内のバッファの設定などの情報を設定する。バッファメモリ18からのデータはVBUSインタフェース部(以下、VBUSIFと記す)163を経由してATMコントローラ16のバケットメモリ164に一旦蓄積される。バケットメモリ164に送られたデータは、SARチップ部(以下、SARCHIPと記す)165によりATMバケットに組み立られ物理層の信号に変換された後、物理層インタフェース部167を介してATM網30に送り出される。

【0045】以下では、マイクロスケジューラ24に関して順次説明していく。まず、本実施形態の概略動作は以下のようなになる。ユーザ端末32から連続データサーバ装置に対して連続データの再生要求が発行されたとき、中央制御装置20はストレージ部4のどのディスク装置12のどの部分に要求された連続データが存在するかを検索し、ディスクアクセスが他のユーザ端末からの要求と競合しないようにディスクアクセスのスケジュールを作成する。

【0046】なお、一般に、ディスク装置12は連続した大きな単位をアクセスするとアクセスの効率がよいの

で、連続データサーバ装置ではデジタル化され圧縮されたデータを比較的大きな単位で（例えば128KByte）バッファメモリ18に読み出し、これを例えば4Mbps程度で読み出す場合には250msec程度かかって再生する。ここで、128KByteのディスクのアクセスは再生の速度250msecより十分速く、例えば60msec程度で終了することに着目すれば、1台のディスク装置で複数の連続データの要求を同時に扱えることがわかる。

【0047】中央制御装置20では、スケジューラと呼ばれるソフトウェアによって複数のユーザからの要求を調停し、ディスク装置12でアクセスの競合がないように効率の良いスケジューリングを行なって多数のユーザからの要求に対してサービスを行なう。制御を簡単にするためにスケジューラはスロットと呼ばれる一定の時間間隔を単位として、例えばスロットの切れ目毎に次のスロットのスケジューリングを決定し、ストレージ部4のSCSIコントローラ14に対して指令を与える。

【0048】ユーザ端末32からの再生要求に基づいて前述のスケジューリングを行った中央制御装置20から指令を与えられたストレージ部4のSCSIコントローラ14は、ディスク装置12から要求されたデータを読み出し、読み出されたデータをバッファメモリ18の指定アドレスへ順次格納する。

【0049】同様に中央制御装置20はスロット単位に端末32にデータを送出するスケジューリングを決定し、ネットワーク部6のATMコントローラ16に対して指令を与え、バッファメモリ18はマイクロスケジューラ24の制御に従ってデータを読み出し、ネットワーク部6はマイクロスケジューラ24の制御に従って、読み出されたデータを取り込む。

【0050】そして、ネットワーク部6からネットワーク30に各要求元のユーザ端末32に宛てて、一定期間ごとに一定量の連続データが送出される。さて、マイクロスケジューラ24は、マイクロスケジュールテーブル22に格納されたマイクロスケジュールのプログラムに従って、バッファメモリ18とネットワーク部6を接続する共通バスVBUSの使用を制御する。すなわち、マイクロスケジューラ24は、ターゲットユニットに対する動作命令を含む所定のフォーマット（図4参照）の制御命令をマイクロスケジュールテーブル22内のテーブルのエントリから1つずつ読み出し、バスの使用を許可するターゲットユニットに与えることにより、制御命令を与えられた該当ユニットだけが共通バスを使用可能としている。なお、図1では、マイクロスケジューラ24は共通バスVBUSとは独立した制御バスにより各ネットワーク部6に指令を与えているが、図3のように共通バスVBUSを使用して各ネットワーク部6に指令を与えるようにしても構わない。ただし、図1の構成の方がより高いバスの使用率を得ることができる。

【0051】上記のマイクロスケジュールのプログラムは、該中央制御装置20によりスケジューリングされ、マイクロスケジュールテーブル22に格納される。マイクロスケジュールテーブル22は、SRAMなどを用いて構成され、中央制御装置20から各スロットに対応して書き換えられる。これはディスク装置12の制御がスロット毎に変更されるのに対応している。

【0052】なお、マイクロスケジュールテーブル22を複数のメモリバンクから構成し、あるバンクへの中央制御装置20からの書き込みと、他のバンクからのマイクロスケジューラ24による読み出しとを並行的に実行することが可能である。

【0053】次に、マイクロスケジュールテーブル22内にマイクロスケジュールテーブルとして保持されるプログラムのフォーマットについて説明する。マイクロスケジュールテーブル22は例えば図4のようなフォーマットの制御命令をテーブル形式で保持している。図4(a)のように、1つのエントリには、少なくとも、命令と読み出し先アドレス（ソース・アドレス）と書き込み先アドレス（ディスティネーション・アドレス）の命令が書き込まれる。本実施形態では詳しくは後述するが図4(b)のように、さらにそのエントリを繰り返す回数に付加して格納する。

【0054】「命令」を表すフィールドにはバッファ18のメモリコントローラ（図示せず）と、ネットワーク部6のATMコントローラ16への命令が格納される。本実施形態では、この命令として次の2種類がある。

(1) 送出命令：バッファメモリからATMコントローラのケットメモリへの転送

(2) 読み込み命令：ATMコントローラからバッファメモリへの転送

その他にも、ストレージシステムにRAIDを採用した場合には、メモリコントローラへRAIDの演算を指示する命令をマイクロスケジューラから送ることが可能である。この場合、中央制御装置がディスク装置の故障を検出し、マイクロスケジュールテーブルにRAIDのための命令を書き込む。

【0055】「送出命令」の場合、読み出し先アドレスはデータの格納されているバッファメモリ18の読み出しの先頭アドレスであり、書き込み先アドレスはターゲットユニットとなるATMコントローラ16内のケットメモリ164のアドレスの情報となる。「読み込み命令」の場合は、その逆となる。

【0056】図5に、マイクロスケジュールテーブルのフォーマットの一例を示す。マイクロスケジュールテーブルは図4に示したマイクロスケジューラの命令を複数個並べた形で構成されている。マイクロスケジューラ24は、これを1エントリずつ順番に実行していく。

【0057】例えば、マイクロスケジューラ24内に次に実行するマイクロスケジュールテーブルのエントリ位



置を保持するポインタカウンタを設け、マイクロスケジューラ24はポインタカウンタの指し示す位置のエントリの内容をもとにバッファメモリ18(のメモリコントローラ)とネットワーク部6(のATMコントローラ16)にデータ転送の指令を出し、データの格納されているバッファメモリ18からターゲットユニットとなるネットワーク部6へデータを送り出させ、あるいはターゲットユニットとなるネットワーク部6からバッファメモリ18にデータを読み込ませる。

【0058】ところで、図5では、マイクロスケジューラテーブルの各エントリに繰り返し回数のフィールドを設けている。繰り返し回数のフィールドは各々のエントリを何回実行してから次のエントリに進むかの情報が示されている。従って、同一のエントリ位置の命令を繰り返している間は、ポインタカウンタの値は更新されないようにする。

【0059】なお、詳しくは後述するが、図7のようにマイクロスケジューラテーブルを複数のバンクに分割し、バンクを切替えながら実行することも可能である。前述のように本実施形態では、中央制御装置20のスケジューラではディスクアクセスのスケジューリングを行なうと同時にマイクロスケジューラテーブルを作成する。このマイクロスケジューラテーブルはスロットに比べて非常に細かい単位で制御される。例えば、本実施形態では4word(16byte)を制御の最小単位とし、このデータを4クロックで読み出すものとしている。この最小単位はバッファメモリの連続読み出しの単位(例えばSynchronousDRAMのバーストサイズ)とすると効率が良い。最小単位は、以降、マイクロスロットと呼ぶ。従って、マイクロスケジューラ24は、マイクロスロット毎にマイクロスケジューラテーブルから情報を読みだし、これを実行していくこととなる。

【0060】ところで、マイクロスケジューラテーブルを1スロット分(例えば60msec)用意すると1クロックが40nsecとして約380Kエントリが必要になり、マイクロスケジューラテーブル用のメモリコストも中央制御装置20からの転送時間も問題になる。そこで、次の方法によりエントリの数を抑えてマイクロスケジューラテーブル用のメモリコストを圧縮することができる。

【0061】第1番目の方法は、マイクロスケジューラテーブルをスロットの分割数分繰り返し使用する方法である。例えば1Kエントリのメモリを用意した場合はこれを380回繰り返し使用するものである。これにより、メモリの容量を削減することができ、上記の例では1/380に減少する。この方法は、マイクロスケジューラ24内にカウンタを設けるなどするだけで容易に実現可能である。

【0062】第2番目の方法は、図5などに示したよう

にマイクロスケジューラテーブルのエントリに繰り返し回数を示すフィールドを設け、一つのエントリを何回か実行することである。

【0063】また、以上の2つの方法を併用すれば、マイクロスケジューラテーブルのメモリ容量を効果的に減らすことができる。ここで、本実施形態におけるマイクロスロット、ミニスロット、スロットの関係について説明する。

【0064】図6は、マイクロスロット、ミニスロット、スロットの関係を示した図である。まず、マイクロスロットは前述のように一定のバスサイクルから構成され、この例では10サイクルでマイクロスロットを構成している。一つのマイクロスロットはマイクロスケジューラテーブルの一つの命令に対応しているので、図6の拡大したマイクロスロットではユーザaへのデータが送られている。

【0065】マイクロスロットが一定の数だけ集まってミニスロットが構成される。図6にはユーザaへのデータ送出とユーザcへのデータ送出だけを示している。この例では、ユーザaへのデータは1ミニスロット内に9マイクロスロットを占めている。このように、ミニスロット内で同一転送命令を複数マイクロスロット連続して実行させる手段は、前述の第2番目の方法により容易に実現可能である。

【0066】スロットは一定の数のミニスロットで構成される。この例では10ミニスロットで1スロットとなっている。ミニスロット1回は、前述の第1番目の方法によればマイクロスケジューラテーブルの一巡である。このミニスロットが繰り返し実行されるので、図6で示すように、スロットよりも細かい単位の一定の間隔でネットワーク部6に連続データが送り込まれ、1スロット内における各ユーザ端末へのデータの転送が一定のレートで行われることを保証することが可能となる。また、データの転送が一定のレートで行われることが保証されると、ATMコントローラ16内部のバケットメモリ164の量を少なくすることが可能である。

【0067】スロットは一定の数のミニスロットで構成される。この例では10ミニスロットで1スロットとなっている。ミニスロット1回は、前述の第1番目の方法によればマイクロスケジューラテーブルの一巡である。このミニスロットが繰り返し実行されるので、図6で示すようにスロット内で各ユーザ端末へのデータの転送が一定のレートで行われることを保証することができる。データの転送が一定のレートで行われることが保証されると、ATMコントローラ16内部のバケットメモリ164やユーザ端末32のバッファの容量を少なくすることができる。

【0068】言い換えると、本実施形態によれば、スロットを複数に等分に分割した各ミニスロット内の一定の位置に、各ユーザ端末宛に送り出されるべきデータをネ

ネットワーク部6に取り込むための共通バスの使用が割り当てられるように、マイクロスケジュールのプログラムを作成することができ、このプログラムに従ってバス使用の制御を行うことにより、スロットよりも細かい単位の一一定の間隔で、ネットワーク部6に連続データが送り込まれることを保証することができる。このようなバス使用の制御は、前述の第1番目の方法と第2番目の方法を併用することにより、容易に実現可能である。

【0069】また、各ユーザ端末宛に送り出されるべきデータのビットレートに応じた数の、スロットを複数に等分に分割したマイクロスロットが、該データをネットワーク部6に取り込むための共通バスの使用期間として割り当てられるように、プログラムを作成し、このプログラムに従ってバス使用の制御を行うことにより、各ユーザ端末の要求ビットレートに応じたバス使用期間を、各スロット内で確保することができる。これを実現する1つの方法は、前記のようなエンタリ内の繰り返し回数に従って実行を繰り返すことにより、ビットレートに応じた使用期間を割り当てるものである。また、図6のようにミニスロットを併用して実現すると、より効果的である。

【0070】次に、マイクロスケジュールテーブルを複数のバンクに分割する例について説明する。図7は、マイクロスケジュールテーブルを2つのバンクに分割した例である。これは図1で示すようにネットワーク部6が2つある場合に対応している。各々のバンクは各々のネットワーク部とのマイクロスケジュールに対応している。

【0071】マイクロスケジュール24は、マイクロスロット毎にBank 0とBank 1を交互に実行する。図8に共通バスVBUSから2つのネットワーク部(0)と(1)への転送のタイミングを示す。

【0072】この方法を使用することによる効果は、ATMコントローラ16のバッファメモリ部164のバンド幅ネックを緩和することにある。例えば、ネットワーク部(0)に連続して要求が行なわれるとネットワーク部(0)のバッファメモリ164は毎クロック書き込みが発生することになり、SAR CHIP165からの要求を受け付けられなくなってATMバッファの送り出しに支障を来たすかもしれない。例えば図9に示すようにVBUSIF163からのデータ転送によりSAR CHIP165はバッファメモリ164にアクセスできない。バッファメモリ164をデュアルポートメモリで構成することも考えられるがコストの増大を招いてしまう。

【0073】そこで、マイクロスケジュール24からの命令を本実施形態のように2つに分けて2つのネットワーク部(0)と(1)に交互に実行させるようにすれば、バッファメモリ164へのバッファメモリ18からの転送は半分になり、バンド幅ネックが緩和される。

【0074】次に、マイクロスケジュールテーブルとバンクとマイクロスロットの関係について説明する。図10は本実施形態で説明する2つのバンクとマイクロスロットの関係を表したものである。図10の一目盛は1マイクロスロットでATMコントローラ(0)への送り出しはBANK 0に、ATMコントローラ(1)への送り出しはBANK 1に記述されている。図10で小文字のアルファベットがそれぞれ異なるユーザ端末への送り出しを示している。ATMコントローラ(0)へはa, a, a, b, b, b, c, c, c, c, cの順で送りだし、ATMコントローラ(1)へはe, f, g, h, i, j, j, j, jの順に送り出している。この例のように共通バスVBUSではATMコントローラ(0)とATMコントローラ(1)へのデータがマイクロスロット毎にインターリーブして使用されている。

【0075】実施形態では二つのバンクに分けた場合を説明したが、これが3または4つ以上のバンクに別れている構成もネットワーク部およびATMコントローラとの関係で可能なことはいうまでもない。

【0076】ところで、マイクロスケジュールテーブルをスロットの分割数分繰り返して使用し、および/または図5や図7のように各エンタリの繰り返し回数を指令し、この情報に従って各エンタリを繰り返し実行する場合、エンタリのソースアドレスおよびディストネーションアドレスのフィールドは繰り返し実行される度に更新される。例えば、マイクロスケジュールテーブルのエンタリを一回実行すると16byteのデータが転送される場合は16byteのインクリメントが行なわれる。このアドレスの更新機能は、マイクロスケジュール24がDMAコントローラとして動作していることを示している。一般にDMAコントローラはハードウェア資源の制約により、限られたチャンネル数のみをサポートしている。例えば2チャンネルのDMAコントローラは、二つの異なるソース/ディストネーションのアドレスペアに対して転送をすることができるが、2チャンネル以上の転送が必要な場合、プロセッサによりDMAの資源をセットし直して転送することが必要になる。

【0077】一方、本実施形態のような連続データサーバ装置を考えた場合、バッファメモリとネットワーク部のバッファメモリへの転送は、連続データサーバ装置のサポートするユーザの数だけチャンネル数が必要で、プロセッサがDMAをセットし直すのはプロセッサにとってかなりの負担になる。

【0078】ここで、マイクロスケジュール24をDMAコントローラとして見ると、サポート可能なチャンネルの数はマイクロスケジュールテーブルのエンタリの数で制限されることになるので、マイクロスケジュールテーブルがメモリで実現されているために、例えば1000チャンネル以上のサポートでも可能になる。マイクロスケジュール24を使えば、DMAのためのセットが高々

1 スロットに1回で良く、スロット内ではDMAのセットし直しをするためにプロセッサが割り込みで処理の中断をされることがない。

【0079】次に、ユーザ端末またはコンテンツの入ったアーカイブ装置からデータをアップロードする場合について説明する。アップロードは、連続データサーバ装置に新たなコンテンツを加える場合に必要な操作である。アップロードは、例えばATM網30にアーカイブ装置から連続データをATMパケットとして送り出しネットワーク部6から読み込む方法や、ネットワーク部6に直接テープデバイスあるいはディスクデバイス等を接続して読み込む方法などにより、共通バスVBUSにデータを流し込むものである。

【0080】アップロードにおいても、読み込まれたデータは同様にマイクロスケジューラ24により確定的に制御されるタイミングで共通バスVBUSに送り出され、共通バスVBUSのデータはバッファメモリ18に書き込まれ、その後バッファメモリ18からストレージ部4に書き込まれる。

【0081】ATM網から受けたデータは、SAR CHIP165によりパケットメモリ164に書き込まれる。パケットメモリ164は、ATM網30と連続データサーバ装置の間のバッファとして働く。パケットメモリ184内部は、例えばリング状のバッファとして管理され、ATM網30から書き込むとライトポインタ(Write Pointer)が進み、リードポインタ(Read Pointer)の方はスロット毎にマイクロスケジューラテーブルで設定しただけネットワークプロセッサ162によって進められる。通常はライトポインタ(Write Pointer)の管理はSAR CHIP165によって行なわれる。

【0082】マイクロスケジューラテーブルには、ATMコントローラ16のパケットメモリ164からバッファメモリ18に書き込む命令を書いておく。この命令によりATMコントローラ16のパケットメモリ164からは共通バスVBUSにデータが送られ、バッファメモリ18は共通バスVBUS上のアップロードデータをバッファメモリ18に書き込む。

【0083】この場合、もしATM網30からのパケットメモリ164へのデータ供給がマイクロスケジューラ24で吸い上げる速度より遅ければ、バッファ領域のリングバッファがemptyになってしまうかもしれない。もし供給が追いつかずemptyになると、マイクロスケジューラ24からの指示に対応するデータが共通バス上に存在しない状態が発生することになるので、ネットワーク部6のネットワークプロセッサ162がempty状態を検出し、マイクロスケジューラ24がパケットメモリ164の読み出しを要求するタイミングで制御信号を出して、共通バスVBUSにはデータを乗せないようにする。また、データが共通バスVBUSに乗

っていないので、マイクロスケジューラ24はエントリ中のアドレスをインクリメントせず、バッファメモリ18はメモリに書き込みを行なわないよう指示する。

【0084】上記の条件は、ATM網30から来るビットレートが一定でない場合で、ビットレートが一定の場合には上記の機構を働かせなくてもパケットメモリ164がバッファとして働き、パケットメモリ164がemptyになることはない。一方、ビットレートが一定でない場合はマイクロスケジューラ24には予想される最大のレートでATM網30からの読み出しの命令を入れておき、前述した機能を働かせると良い。

【0085】以上、一例として、ATM網からVBUS経由でアップロードデータ書き込む方法を示したが、VBUSに直接データを供給するデバイスをつけることが可能であることはいうまでもない。

【0086】次に、マイクロスケジューラ24およびマイクロスケジューラテーブル22の構成について説明する。図11に、マイクロスケジューラ24の内部構成およびマイクロスケジューラテーブル22の構成の一例を示す。

【0087】マイクロスケジューラテーブル22にはSRAMを使用し、中央制御装置20から書き込めるようになっている。図11の構成では、SRAM22がネットワーク部16の数に合わせて二つのバンクBank0とBank1分けられており(図7参照)、各メモリバンクにはそれぞれ対応するターゲットユニットへの指示の基となるプログラムが格納される。これに対応して、後述するアドレスカウンタおよび繰り返しカウンタとも、2組ずつ存在する。これらを切替えるのは図11中に示したBANK信号である。BANK信号は、1マイクロスロットごとに1と0の反転を繰り返す信号である。これらによって、ターゲットユニットのそれぞれに対する指示を1マイクロスロットごとにインターリーブして転送できるようにしている。

【0088】マイクロスケジューラテーブル22からマイクロスケジューラ24内に読み出されたデータは一旦、レジスタ241に蓄えられる。これとともに、命令のフィールドの内容が「送出命令」の場合には、バッファメモリ18にソースアドレスと命令が、ATMコントローラ16にはディスティネーションアドレスと命令が各々転送され、「読み込み命令」の場合には、バッファメモリ18にディスティネーションアドレスと命令が、ATMコントローラ16にはソースアドレスと命令が各々転送される。

【0089】ソースアドレスおよびディスティネーションアドレスは、それぞれインクリメント器243、242により、1マイクロスロットで転送されるデータ量に相当するバッファのアドレス分だけインクリメントされた後、書き込みゲート244によりマイクロスケジューラテーブル22に再び書き戻される。

【0090】CONTビットは、初期値が1にセットされ、このときレジスタ241の繰り返し回数データが繰り返しカウンタ248または250にロードされ、CONTビットには0が書き戻される。繰り返しカウンタ248または250の内容は、該当エントリが実行される度に1づつデクリメントされ、繰り返しカウンタ248または250が0になったら、次に実行するマイクロスケジュールテーブルのエントリ位置を指し示すポインタカウンタ251または252をインクリメントするとともに、再びCONTビットに1が書き戻される。

【0091】図11では、SRAM22はデュアルポート構成で、中央制御装置20とマイクロスケジューラ24の両方からアクセスできるようになっている。SRAM22をデュアルポート構成ではなく、ダブルバッファ構成にすることも可能である。この場合、中央制御装置20が一方のバンクに書き込んでいる間、もう一方のバンクをマイクロスケジューラが読み出す。

【0092】マイクロスケジュールテーブルの繰り返し回数により、ビデオサーバ装置では異なったデータレートのビデオを混在させることが可能である。例えば、VBUSのデータ転送速度を33MHz 32bitで1056Mbit/secであるとき、マイクロスケジューラのエントリ数が4096として繰り返し回数1では $1056\text{Mbit}/4096 = 258\text{Kbit}/\text{sec}$ となる。4Mbit/secの転送には繰り返し回数を16にする。このように繰り返し回数を設定することで容易にビデオ転送レートを設定することが可能である。

【0093】転送レートには258Kbitの量子化誤差が存在することになるが、端末32や、ATMコントローラ16にバッファがあるので、この量子化誤差はスロット毎に繰り返し回数を調整することで一定の範囲内に収めることができる。

【0094】マイクロスケジューラ24を使った場合の効果の一つは、高度にパイプライン化されたアクセスによるバスの効率的利用である。マイクロスケジューラ24を用いた場合、図12に示されるようにバッファメモリ18からネットワーク部6に読み出されるデータのアクセス時間を隠蔽でき、かつデータを読み出しながらデータの転送が可能となる。

【0095】通常の方式では、複数のリクエストの調停に時間がかかり、しかもデータの転送と読み出しを並行して行なえないので全ての処理が直列に行なわれるが、マイクロスケジューラ方式では図12のように並行して処理を行なうことでバスの使用効率が極めて高い。

【0096】(実施形態2)次に、実施形態2について説明する。図13に本実施形態に係る連続データサーバ装置の構成を示す。

【0097】本実施形態の連続データサーバ装置は、連続データを格納している所定台数のデータ記憶装置12

と該データ記憶装置12からデータを読み出すデータ記憶制御装置14とからなる、複数(ここでは4系統とする)のストレージ部4、該ストレージ部4から読み出したデータを一時記憶する、ストレージ部4に対応して設けられたバッファメモリ18、該バッファメモリ18のデータを各端末32を宛先としてネットワーク30に送り出す複数(ここでは2台とする)のネットワーク部6、ストレージ部4とバッファメモリ18とネットワーク部6を接続する共通バスVBUS、端末32からの要求に基づいてデータ記憶装置12からデータを読み出してネットワーク30に送り出すまでのスケジューリング、ネットワークの設定をはじめとして、システム全体の制御を行なう中央制御装置20、該中央制御装置20により確定されたマイクロスケジュールのプログラムを格納するマイクロスケジュールテーブル22、該プログラムに従ってネットワーク部6に共通バスVBUSを使用する権利を確定的に割り当てるマイクロスケジューラ24を備えている。

【0098】すなわち、本実施形態は、実施形態1のストレージ部4およびバッファメモリ18を複数系統(図13では4系統)分、共通バスVBUSに接続し、各バッファメモリ18を各ネットワーク部6のATMコントローラ16に接続したものである。

【0099】本実施形態のように複数系列のストレージ部4を設けた場合、ディスク装置12にデータをストライピングして配置し、ディスクを並列に読み出すことにより、ディスクアクセスのバンド幅を大きくでき、より多くのユーザに同時に連続データをサービスすることができる。また、複数系列あるストレージ部4の一部をパリティ符号として割り当てることにより、RAIDシステムを構成することができる。

【0100】図14は、各バッファメモリ18からネットワーク部6にデータを送り出すときのマイクロスケジューラからの命令とバッファメモリ18からネットワーク部6に送り出されるデータの関係を示したものである。タイミング1でマイクロスケジューラ24からバッファメモリ18の読み出し要求1が各バッファメモリ18に出されると、時刻6でバッファメモリ(0)からデータが出力され、時刻7でバッファメモリ(1)からデータが出力され、時刻8でバッファメモリ(2)からデータが出力され、時刻9でバッファメモリ(3)からデータが出力されるというように、順次4台のバッファメモリ18からデータが読み出されている。ここで重要な点は、時刻1の読みだし要求1に対する応答が時刻6から始まっていることで、パイプライン動作によりメモリのレーテンシを隠蔽している。

【0101】本実施形態ではマイクロスケジューラ24からバッファメモリ18およびATMコントローラ16への制御バスを専用で設けているが、実施形態1と同様に、マイクロスケジューラ24から専用のバスを設けず

に、共通VBUSを共用する方法もある。

【0102】(実施形態3)次に、実施形態3について説明する。実施形態2ではATMコントローラ16とバッファメモリ18間のバスをマイクロスケジューラ24で制御していたが、マイクロスケジューラ24でSCSIコントローラ14とバッファメモリ18の間のバスを制御することもできる。

【0103】図15に本実施形態に係る連続データサーバ装置の構成を示す。本実施形態の連続サーバ装置は、基本的には実施形態2と同じ構成であるが、マイクロスケジューラ24がストレージ側とネットワーク側の両方についてバス使用の制御を行う点、マイクロスケジューラ24が格納するマイクロスケジュールのプログラムをストレージ側とネットワーク側の夫々について作成する点が相違する。

【0104】本実施形態のように、マイクロスケジューラ24でストレージ側とネットワーク側の両方を制御する場合は、マイクロスケジューラ24およびマイクロスケジュールテーブル22の構成は次の2つの場合が考えられる。

【0105】(1) 図16で示すように、ストレージ側を制御するマイクロスケジューラ24およびマイクロスケジュールテーブル22とネットワーク側を制御するマイクロスケジューラ24およびマイクロスケジュールテーブル22を独立に構成する。

【0106】(2) 図17で示すように、ストレージ側を制御するマイクロスケジューラ24とネットワーク側を制御するマイクロスケジューラ24は共通で、マイクロスケジュール管理テーブル22がストレージ側を制御する部分と、ネットワーク側を制御する部分の2つに分かれている。

【0107】また、図18に示すように、SCSIコントローラ14の内部構成は、ネットワーク部6のATMコントローラ16の内部構成(図2)のSARCHIP165をSCSI CHIP185に置き換え、バッファメモリ164をストレージメモリ184に置き換え、外部へのATM網をSCSIに置き換えられたものとなる。

【0108】本実施形態では、ネットワーク側の部分については、実施形態1あるいは実施形態2と同様であるが、ストレージ側については、ストレージ部4から読み出されたデータをバッファメモリ18に取り込む際の共通バスの使用を割り当てるマイクロスケジュールのプログラムを作成する。

【0109】本実施形態では、まず、ディスク装置12から読み出されたデータはSCSIコントローラ14のストレージメモリ184に蓄積される。SCSIコントローラ14のストレージメモリ184の容量は、接続されたディスク装置12に1スロットでアクセスされるデータサイズより大きい。例えば、1スロットで64KB

byteアクセスされるディスク装置12が1スロットで4つに対して読み出し命令をSCSIコントローラ14から発行する場合は、ストレージメモリ184の容量は、256KByteとなる。

【0110】以下では、インプリメントが最も容易であるダブルバッファをストレージメモリ184に使用した場合について説明する。あるスロットでストレージメモリ184に読み出されたデータは、次のスロットでバッファメモリ18に転送される。ダブルバッファを使っているため、バッファメモリ18に転送している間、もう一方のバンクはディスク装置12からストレージメモリ184への転送を行っていることになる。ストレージメモリ184とバッファメモリ18間の転送は一定の速度で、ほぼ100%バスを使っている転送が可能である。この場合、ストレージメモリ184はディスク装置12から一定の速度で出てこないデータをバッファリングする働きを行う。

【0111】一方、前述のようにコンテンツをロードする場合、このバスは全く逆の方向に働く。すなわち、バッファメモリ18からストレージメモリ184にマイクロスケジューラ24の指示によりデータが転送され、次にストレージメモリ284からディスク装置12にデータが転送される。この場合にも、ストレージメモリ184はバッファとして働く。

【0112】なお、本実施形態では、マイクロスケジューラ24がストレージ側だけについてバス使用の制御を行う実施形態も可能である。この場合、上記構成において、マイクロスケジューラ24からネットワーク部6へ情報を伝える手段が不要となり、マイクロスケジューラ24およびマイクロスケジュールテーブル22はストレージ側対応に設けられ、中央制御装置20はストレージ側に対するマイクロスケジュールリングのみ行うように修正する。また、この場合、ストレージ側についての構成・動作は上記と同様であり、ネットワーク側についての構成・動作は、従来技術と同様になる(共通バスの使用権は調停装置により制御される)。

【0113】以上の各実施形態ではSCSIを使用してディスクを接続しているが、ファイバチャネル等他のディスクインタフェースが使用できるのはいうまでもない。また、ネットワーク部にATMを使用しているが、イーサネットなど他のネットワークを用いるのも好ましい実施形態の一例である。また、以上の実施形態ではマイクロスケジューラはシステムに1個であったがバッファメモリ毎に設けることも可能である。本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【0114】

【発明の効果】本発明(請求項1)に係る計算機装置によれば、各ユニットからのバス使用要求に依存してバス使用権の調停を行うのではなく、格納手段に格納された

プログラムに従って割当手段により、各ユニットが共通バスを使用する権利を決定論的に割り当てるようにしたので、予め定めたスケジューリングによりバス使用权を割り当てるため、各ユニットのバス使用時期を確定的に保証することができる。このため、メモリへの指示を先行発行することができるので、メモリのレーテンシを隠蔽することができる。従って、バスの使用率の向上を図ることができる。

【0115】本発明（請求項9）に係る連続データサーバによれば、バッファメモリユニットと通信制御ユニットの間でのデータ転送に共通バスを使用する権利を、プログラムに従って各通信制御ユニットに対し決定論的に割り当てるようにしたので、請求項1の発明の効果に加え、一定の間隔で通信制御ユニットに連続データが送り込まれ、ひいては、一定の間隔で通信路に連続データが送り出されることを保証することができる。従って、通信制御ユニットが通信路に連続データを送り出すまで該データを一時記憶するメモリの容量を少なくでき、また、通信路の先に存在して連続データを再生するユーザ端末において受信したパケットを保持するバッファの容量を少なくできる。

【図面の簡単な説明】

【図1】本発明の実施形態1に係る連続データサーバ装置の構成を示す図

【図2】ネットワーク部の内部構成の一例を示す図

【図3】同実施形態に係る連続データサーバ装置の他の構成を示す図

【図4】マイクロスケジュールテーブル22に格納される単位プログラムのフォーマットの一例を示す図

【図5】マイクロスケジュールテーブルのフォーマットの一例を示す図

【図6】マイクロスロット、ミニスロット、スロットの関係の説明するための図

【図7】マイクロスケジュールテーブルのフォーマットの他の例を示す図

【図8】バンク切替えによる2つのネットワーク部への転送のタイミングを示すタイミングチャート

【図9】ネットワーク部内でのデータの流れを説明するための図

【図10】2つのバンクとマイクロスロットの関係を説明するための図

【図11】マイクロスケジューラの内部構成の一例およびマイクロスケジュールテーブルの構成の一例を示す図

【図12】同実施形態におけるバイブライン処理を説明するための図

【図13】本発明の実施形態2に係る連続データサーバ装置の構成を示す図

【図14】各バッファメモリからネットワーク部にデータを送り出すときのマイクロスケジューラからの命令とバッファメモリからネットワーク部に送り出されるデータの関係を説明するための図

【図15】本発明の実施形態3に係る連続データサーバ装置の構成を示す図

【図16】同実施形態におけるマイクロスケジューラおよびマイクロスケジュールテーブルの構成の一例を示す図

【図17】同実施形態におけるマイクロスケジューラおよびマイクロスケジュールテーブルの構成の他の例を示す図

【図18】同実施形態におけるSCSIコントローラ14の内部構成の一例を示す図

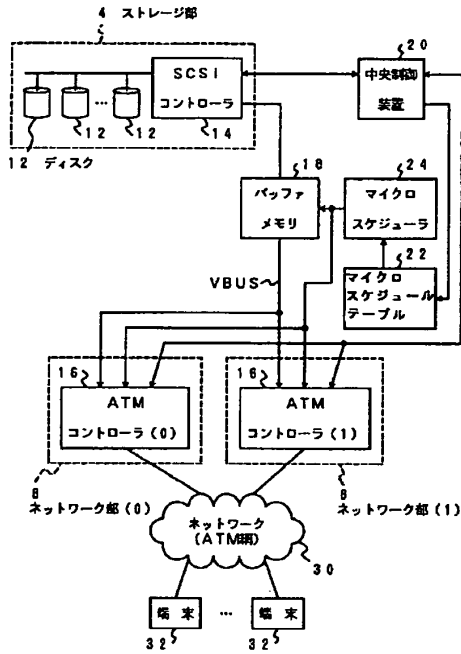
【図19】従来の連続データサーバ装置の構成の一例を示す図

【図20】従来の連続データサーバ装置の構成の他の例を示す図

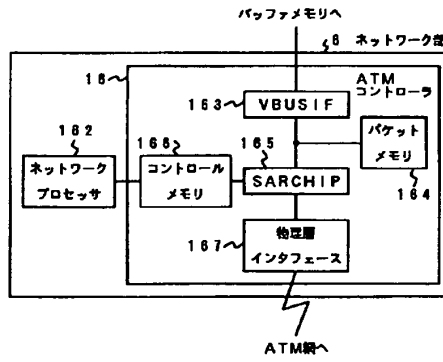
【符号の説明】

4…ストレージ部  
12…ディスク装置  
14…SCSIコントローラ  
6…ネットワーク部  
16…ATMコントローラ  
162…ネットワークプロセッサ  
163…VBIUSIF  
164…パケットメモリ  
165…SAR CHIP  
166…コントロール・メモリ  
167…物理層インタフェース部  
18…バッファメモリ  
20…中央制御装置  
22…マイクロスケジュールテーブル  
24…マイクロスケジューラ  
VBUS…共通バス  
30…ネットワーク  
32…ユーザ端末  
241…レジスタ  
242, 243…インクリメント器  
244…書き込みゲート  
245, 253…マルチプレクサ  
246, 247, 249…論理演算素子  
248, 250…繰り返しカウンタ  
251, 252…ポインタカウンタ  
185…SCSI CHIP  
184…ストレージメモリ

【図1】



【図2】



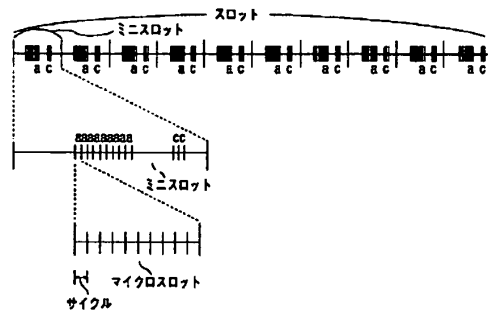
【図4】

- (a) 命令 | ソース・アドレス | ディスティネーション・アドレス
- (b) 命令 | ソース・アドレス | ディスティネーション・アドレス | 繰り返し回数

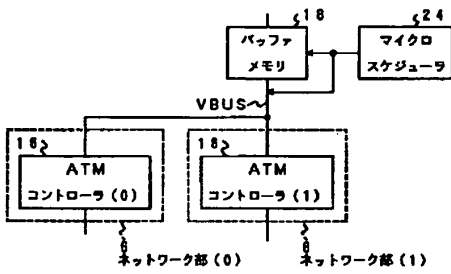
【図5】

命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ1
命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ2
:				:
命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリi
:				:

【図6】



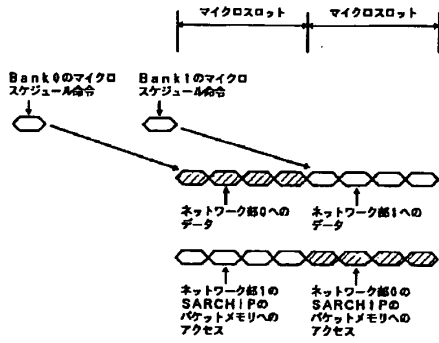
【図3】



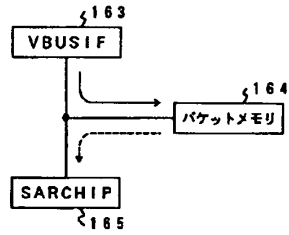
【図7】

Bank 0	命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ1
	命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリ2
:					:
Bank 1	命令	ソース・アドレス	ディスティネーション・アドレス	繰り返し回数	エントリi
	:				

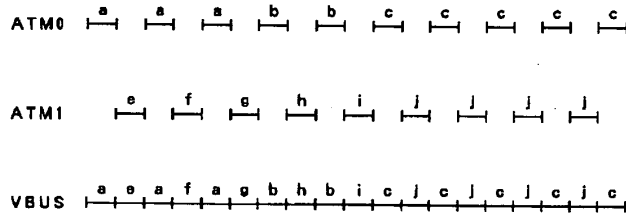
【図8】



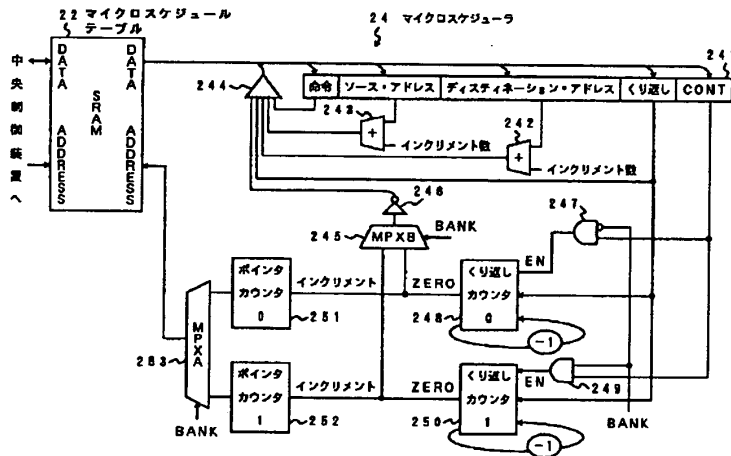
【図9】



【図10】

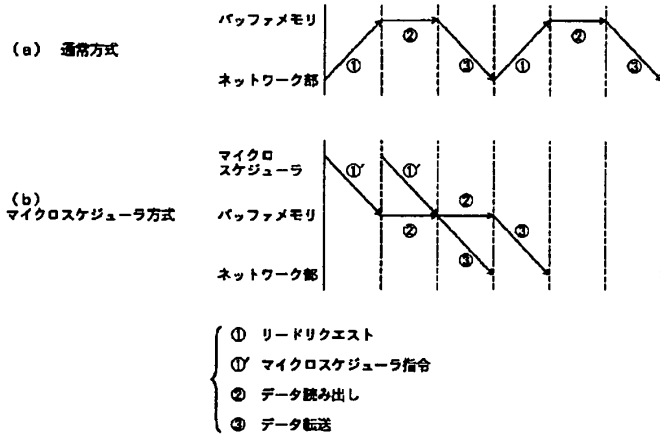


【図11】

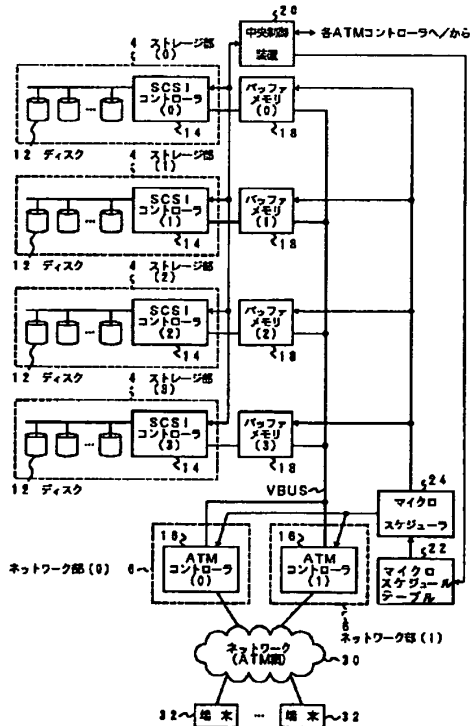




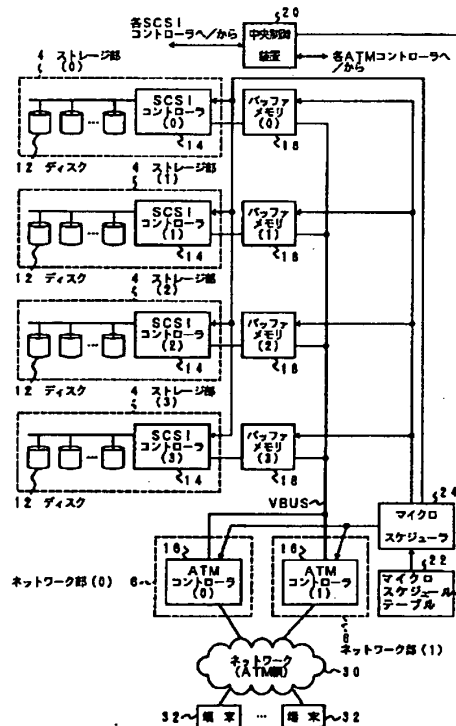
【図12】



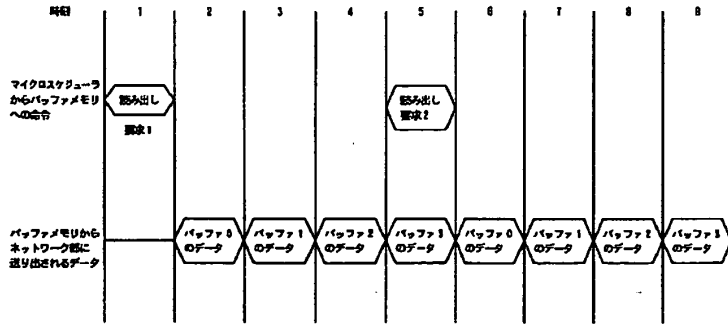
【図13】



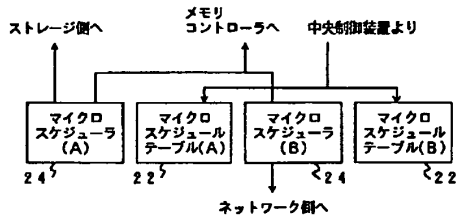
【図15】



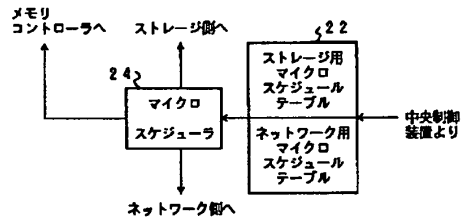
【図14】



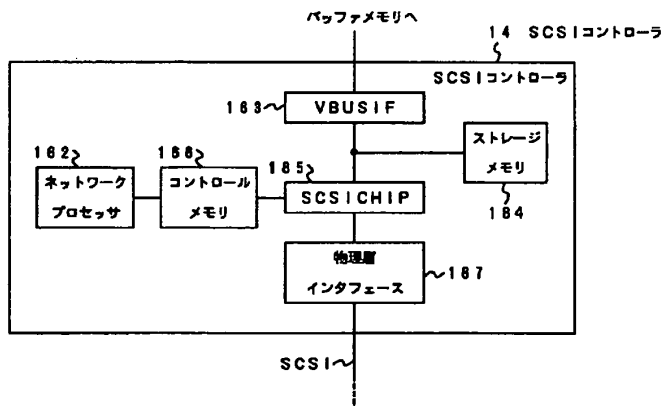
【図16】



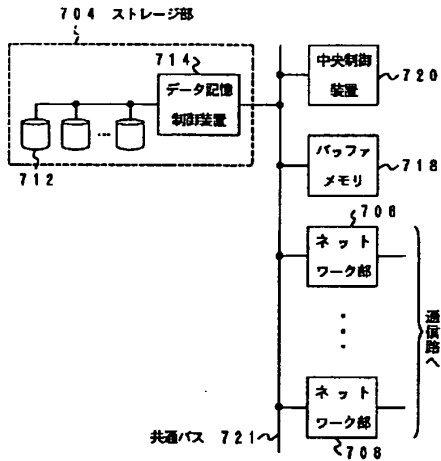
【図17】



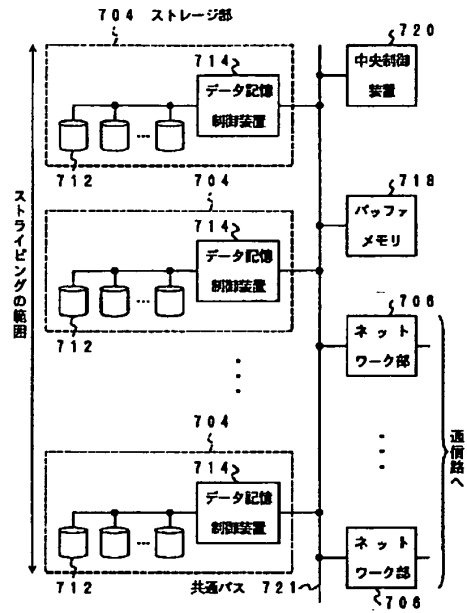
【図18】



【図19】



【図20】



B 20

JP1993181609A

1993-7-23

**Bibliographic Fields**

**Document Identity**

(19)【発行国】 日本国特許庁(JP)	(19) [Publication Office] Japan Patent Office (JP)
(12)【公報種別】 公開特許公報(A)	(12) [Kind of Document] Unexamined Patent Publication (A)
(11)【公開番号】 特開平5-181609	(11) [Publication Number of Unexamined Application] Japan Unexamined Patent Publication Hei 5- 181609
(43)【公開日】 平成5年(1993)7月23日	(43) [Publication Date of Unexamined Application] 1993 (1993) July 23*

**Public Availability**

(43)【公開日】 平成5年(1993)7月23日	(43) [Publication Date of Unexamined Application] 1993 (1993) July 23*
------------------------------	---

**Technical**

(54)【発明の名称】 パーソナルコンピュータシステム	(54) [Title of Invention] <b>PERSONAL COMPUTER SYSTEM</b>
(51)【国際特許分類第5版】 G06F 3/06 301 Z 7165-5B	(51) [International Patent Classification, 5th Edition] G06F3/06301Z7165-5B
【請求項の数】 1	[Number of Claims] 1
【全頁数】 4	[Number of Pages in Document] 4

**Filing**

【審査請求】 未請求	[Request for Examination] Unrequested
(21)【出願番号】 特願平4-333	(21) [Application Number] Japan Patent Application Hei 4- 333
(22)【出願日】 平成4年(1992)1月6日	(22) [Application Date] 1992 (1992) January 6*

**Parties**

**Applicants**

(71)【出願人】	(71) [Applicant]
【識別番号】 000004237	[Identification Number] 000004237
【氏名又は名称】	[Name]

JP1993181609A

1993-7-23

日本電気株式会社

NEC CORPORATION (DB 69-054-1685)

【住所又は居所】

[Address]

東京都港区芝五丁目7番1号

Tokyo Minato-ku grass 5-7-1

**Inventors**

(72)【発明者】

(72) [Inventor]

【氏名】

[Name]

平井 秀生

Hirai Hideo

【住所又は居所】

[Address]

東京都港区芝五丁目7番1号 日本電気株式会社内

Tokyo Minato-ku grass 5-7-1 NEC Corporation (DB 69-054-1685) \*

**Agents**

(74)【代理人】

(74) [Attorney(s) Representing All Applicants]

【弁理士】

[Patent Attorney]

【氏名又は名称】

[Name]

若林 忠

Wakabayashi Tadashi

**Abstract**

(57)【要約】

(57) [Abstract ]

【目的】

[Objective ]

複数の磁気ディスク装置を複数のパーソナルコンピュータで共用できるパーソナルコンピュータシステムを提供する。

personal computer system which can share magnetic disk device of plural with personal computer of the plural is offered.

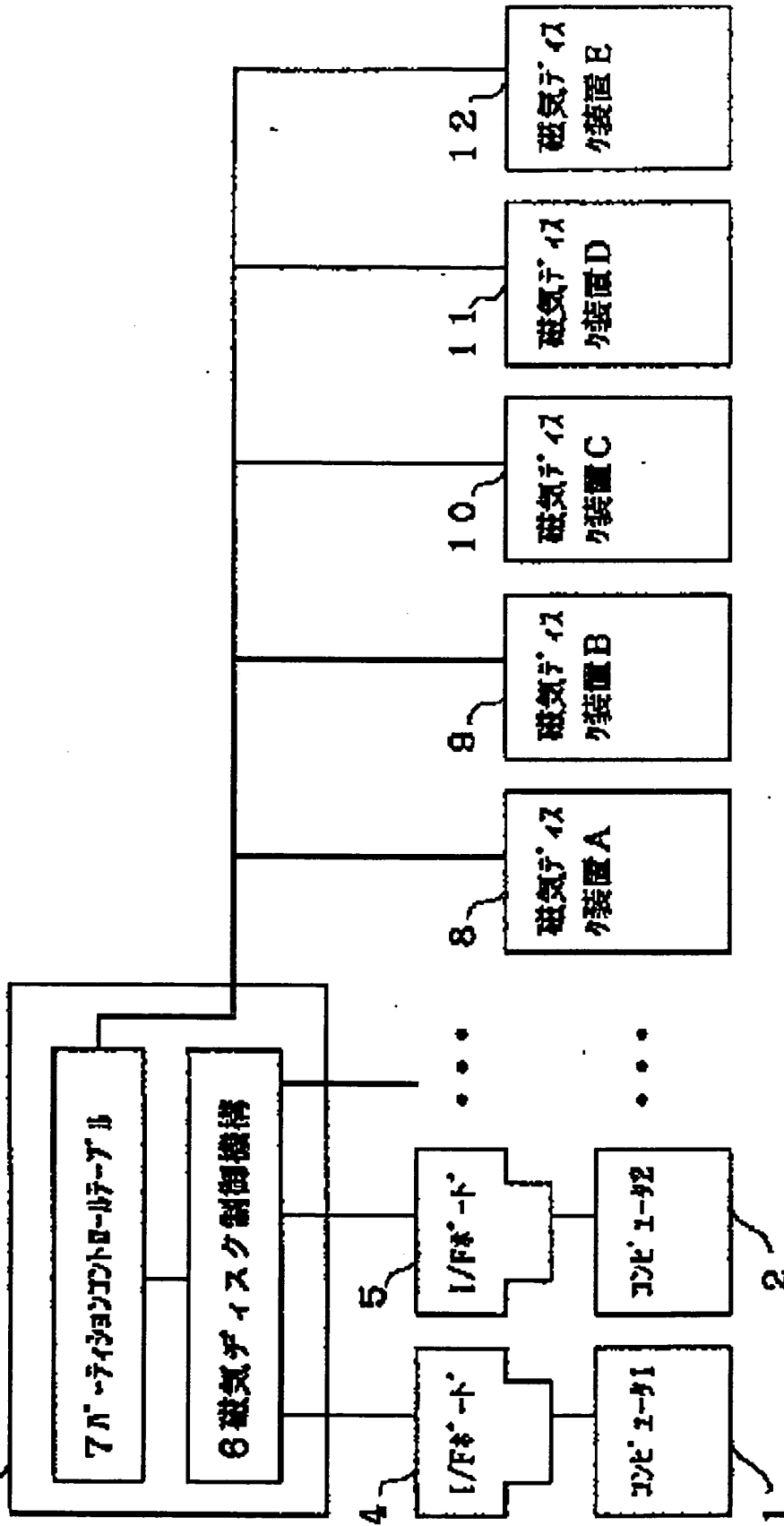
【構成】

[Constitution ]

複数の磁気ディスク装置 8~12 を、それらの全記憶領域を記憶領域とする 1 個の仮想磁気ディスク装置とみなして制御する磁気ディスク制御機構 6 と、仮想磁気ディスク装置の記憶領域のパーティションごとに指定する各パーソナルコンピュータ 1, 2, ... のアクセス権を管理するパーティション・コントロール・テーブル 7 とを備えた磁気ディスク共有装置 3 を有し、各パーソナルコンピュータはそれぞれのアクセス権にしたがって仮想磁気ディスク装置にアクセスする。

Regarding magnetic disk device 8~12 of plural , hypothetical magnetic disk device of 1 those all storage area are designated as storage area , magnetic disk joint ownership device 3 which has partition \* control & table 7 which manage access privilege of each personal computer 1, 2, \*\*\* which it appoints every partition of storage area of magnetic disk control mechanism 6 and the hypothetical magnetic disk device which it controls possessing, Following to respective access privilege , access it designates each personal computer as hypothetical magnetic disk device .

3 磁気ディスク共有装置



5-7-23

7,296)

## Claims

## 【特許請求の範囲】

## 【請求項 1】

複数のパーソナルコンピュータと複数の磁気ディスク装置を含むパーソナルコンピュータシステムにおいて、

前記複数の磁気ディスク装置を、該複数の磁気ディスク装置の全記憶領域をその記憶領域とする 1 個の仮想磁気ディスク装置とみなして制御する磁気ディスク装置の制御手段と、当該仮想磁気ディスク装置の記憶領域における前記複数のパーソナルコンピュータそれぞれの利用可能な権利を管理するセキュリティ管理手段とを備えた磁気ディスク共有装置を有し、

前記複数のパーソナルコンピュータはそれぞれ利用可能な前記権利にしたがって前記仮想磁気ディスク装置にアクセスすることを特徴とするパーソナルコンピュータシステム。

## Specification

## 【発明の詳細な説明】

## 【0001】

## 【産業上の利用分野】

本発明は、複数のパーソナルコンピュータと複数の磁気ディスク装置を含むパーソナルコンピュータシステムに関する。

## 【0002】

## 【従来の技術】

従来、パーソナルコンピュータシステムにおける磁気ディスク制御装置は 1 個の論理磁気ディスク装置に対して 1 個の物理磁気ディスク装置しか対応できず、上位ソフトウェア(オペレーティングシステム)の磁気ディスク管理も同様となっている。

## 【0003】

また、磁気ディスク装置を複数のパーソナルコンピュータで共用できなかった。

## 【0004】

## 【発明が解決しようとする課題】

上述した従来の磁気ディスク制御装置は、1 個の論理磁気ディスク装置に 1 個の物理磁気ディスク装置しか対応できない仕様となっているため、1 個の磁気ディスク装置の容量を超えるデータを扱うことができないという欠点があり、ま

## [Claim (s)]

## [Claim 1]

In personal computer of plural and personal computer system which includes magnetic disk device of plural ,

Regarding magnetic disk device of aforementioned plural , hypothetical magnetic disk device of 1 all storage area of magnetic disk device of said plural is designated as the storage area , control means of magnetic disk device which it controls and personal computer respective practical right of aforementioned plural in storage area of the this said hypothetical magnetic disk device magnetic disk joint ownership device which has the security administration means which manages possessing,

As for personal computer of aforementioned plural following to therespective practical aforementioned right, personal computer system . which designates that access it makes aforementioned hypothetical magnetic disk device as feature

## [Description of the Invention]

## [0001]

## [Field of Industrial Application]

this invention regards personal computer of plural and personal computer system which includes magnetic disk device of plural .

## [0002]

## [Prior Art]

Until recently, only physical magnetic disk device of 1 be able to correspond magnetic disk controller in personal computer system vis-a-vis logic magnetic disk device of 1, also magnetic disk management of upper position software (operating system ) has become similar.

## [0003]

In addition, magnetic disk device could not be shared with personal computer of the plural .

## [0004]

## [Problems to be Solved by the Invention]

As for conventional magnetic disk control device which description above is done, because itbecomes specification which only physical magnetic disk device of 1 it can correspond to the logic magnetic disk device of 1, it is not possible, to handle data which exceeds the capacity of

た磁気ディスク装置を複数のパーソナルコンピュータによって共用することができないという欠点があった。

【0005】

本発明の目的は、複数の磁気ディスク装置の全記憶領域をあたかも 1 個の磁気ディスク装置の記憶領域として個々のパーソナルコンピュータからアクセスすることにより、一台の磁気ディスク装置の容量を超える大きさのデータを扱うことを可能にするとともに、個々のパーソナルコンピュータのアクセス権を管理しつつ、複数のマイクロコンピュータによって複数の磁気ディスク装置を共用することのできる磁気ディスク制御装置を有するマイクロコンピュータシステムを提供することである。

【0006】

【課題を解決するための手段】

本発明のパーソナルコンピュータシステムは、複数の磁気ディスク装置を、それら複数の磁気ディスク装置の全記憶領域をその記憶領域とする 1 個の仮想磁気ディスク装置とみなして制御する磁気ディスク装置の制御手段と、その仮想磁気ディスク装置の記憶領域における複数のパーソナルコンピュータそれぞれの利用可能な権利を管理するセキュリティ管理手段とを備えた磁気ディスク共有装置を有し、複数のパーソナルコンピュータはそれぞれ利用可能な権利にしたがって仮想磁気ディスク装置にアクセスする。

【0007】

【作用】

個々のパーソナルコンピュータにとり、複数の磁気ディスク装置の全記憶領域をあたかも 1 個の磁気ディスク装置の記憶領域であるかのようにアクセスすることを可能とし、その 1 個の仮想磁気ディスク装置の記憶領域に対する複数のパーソナルコンピュータそれぞれの利用可能な権利は個々のマイクロコンピュータごとに指定され管理される。

【0008】

【実施例】

次に、本発明の実施例について図面を参照して説明する。

【0009】

図 1 は本発明の、磁気ディスク共用装置を含むパーソナルコンピュータシステムの一実施例の

magnetic disk device of 1 deficiency that is \* there is a deficiency that in addition cannot share magnetic disk device with personal computer of plural .

【0005】

As for object of this invention , as it makes that data of size which exceeds capacity of magnetic disk device of single platform by access doing from individual personal computer with all storage area of magnetic disk device of plural as storage area of magnetic disk device of 1 just, is handled possible, while managing access privilege of individual personal computer , It is to offer microcomputer system which possesses magnetic disk controller which can share the magnetic disk device of plural with microcomputer of plural .

【0006】

【Means to Solve the Problems】

As for personal computer system of this invention , regarding magnetic disk device of plural , the hypothetical magnetic disk device of 1 all storage area of magnetic disk device of those plural is designated as storage area , control means of magnetic disk device which it controls and personal computer respective practical right of plural in storage area of the hypothetical magnetic disk device magnetic disk joint ownership device which has security administration means which manages possessing, Following to respective practical right, access it designates the personal computer of plural as hypothetical magnetic disk device .

【0007】

【Working Principle】

You take in individual personal computer , all storage area of magnetic disk device of plural like whether it is a storage area of magnetic disk device of 1 just of, it makes that access it does possible, personal computer respective practical right of plural for storage area of hypothetical magnetic disk device of 1 of that is appointed every individual microcomputer and is managed.

【0008】

【Working Example (s)】

Next, referring to drawing concerning Working Example of this invention , you explain.

【0009】

Figure 1 is block diagram which shows configuration of one Working Example of personal computer system which



構成を示すブロック図である。

[0010]

図 1 において、パーソナルコンピュータ本体 1, 2, ... はインタフェースボード 4, 5, ... を介して磁気ディスク共用装置 3 に接続されている。

また、磁気ディスク共用装置 3 には磁気ディスク装置 8~12 が接続されている。

磁気ディスク共用装置 3 は、磁気ディスク制御機構 6 とパーティション・コントロール・テーブル 7 から構成されている。

[0011]

パーソナルコンピュータ 1, 2, ... から磁気ディスク装置 8~12 へのアクセス要求は、磁気ディスク用インタフェースボード 4, 5, ... を通じ磁気ディスク制御機構 6 に通知され、磁気ディスク制御機構 6 において磁気ディスク装置 8~12 にまたがる仮想的な磁気ディスク装置に対するアクセス要求に変換される。

以上の処理によりパーソナルコンピュータ本体からは、磁気ディスク装置 8~12 を、磁気ディスク装置 8~12 の全記憶領域を自らの記録領域とする仮想化された一つの磁気ディスク装置として扱うことが可能となる。

[0012]

セキュリティ管理は、上述の仮想磁気ディスク装置の記憶領域を区分けし、区分けされた各部分(パーティションと云う)に、個々のパーソナルコンピュータごとの利用可能な権利を設定し、不正なアクセスを防ぐためのもので、パーティション・コントロール・テーブル 7 を作成して行われる。

パーティションへのアクセス権には、R(読み出し)、W(書き込み)、C(作成)、D(消去)、X(実行)がある。

[0013]

図 2 はパーティション・コントロール・テーブルの例である。

パーソナルコンピュータ 1 は、パーティション 1 に対し読み出し、書き込み、作成、実行が可能であり、パーティション 2 に対し読み出し、書き込みが可能であり、パーティション 3 に対し読み出しが可能である。

パーソナルコンピュータ 2 は、パーティション 1 に対し読み出し、書き込み、作成、実行が可能であり、パーティション 3 に対し読み出しが可能である。

includes, magnetic disk common device of this invention .

[0010]

In Figure 1 , personal computer main body 1, 2, ... through interface board 4, 5, ... , is connected to magnetic disk common device 3.

In addition, magnetic disk device 8~12 is connected to magnetic disk common device 3.

magnetic disk common device 3 configuration is done from magnetic disk control mechanism 6 and partition \* control & table 7.

[0011]

access demand for magnetic disk device 8~12 is notified by magnetic disk control mechanism 6 from personal computer 1, 2, ... via interface board 4, 5, ... for magnetic disk , is converted to access request for the imaginary magnetic disk device which extends over magnetic disk device 8~12 in magnetic disk control mechanism 6.

It becomes possible from personal computer main body depending upon treatment above to handle magnetic disk device 8~12, all storage area of magnetic disk device 8~12 is designated as recording region of self as magnetic disk device of one which is imagined is converted.

[0012]

security management separation does storage area of above-mentioned hypothetical magnetic disk device , sets practical right every of individual personal computer to each portion (You call partition ) which separation is done, with those in order to prevent illegitimate access , drawing up partition \* control & table 7, is done.

R (reading ) , W (writing ) , C (Compilation) , D (Elimination) , there is a X (Execution) in access privilege to partition .

[0013]

Figure 2 is example of partition \* control & table .

As for personal computer 1, reading , writing , compilation and execution being possible vis-a-vis partition 1, reading , writing being possible [paateishon ] vis-a-vis 2, the reading is possible [paateishon ] vis-a-vis 3.

As for personal computer 2, reading , writing , compilation and execution being possible [paateishon ] vis-a-vis 1, reading is possible vis-a-vis partition 3.

パーソナルコンピュータ3は、パーティション1に対し読み出し、書き込み、作成、実行が可能であり、パーティション2に対し読み出しが可能である。

上記セキュリティ管理手段により、パーソナルコンピュータからの利用の許されていない不正なアクセスを防止することが可能となる。

[0014]

[発明の効果]

以上説明したように、本発明の磁気ディスク共用装置を有するパーソナルコンピュータシステムは、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域として個々のパーソナルコンピュータからアクセスし、かつ各パーソナルコンピュータのアクセス権を管理することにより、1個の磁気ディスク装置の記憶容量を超える大きさのデータを扱うことが可能になるとともに、複数のマイクロコンピュータにより複数の磁気ディスク装置を共用することを可能にし、データを個々のパーソナルコンピュータで保管することなく一括して管理することができる効果がある。

[図面の簡単な説明]

[図1]

本発明のパーソナルコンピュータシステムの一実施例である。

[図2]

パーティション・コントロール・テーブルの例である。

[符号の説明]

- 1  
パーソナルコンピュータ本体 1
- 10  
磁気ディスク装置
- 11  
磁気ディスク装置
- 12  
磁気ディスク装置
- 2  
パーソナルコンピュータ本体 2
- 3

As for personal computer 3, reading, writing, compilation and execution being possible vis-a-vis partition 1, reading is possible vis-a-vis partition 2.

By above-mentioned security administration means, it becomes possible to prevent the illegitimate access where utilization from personal computer is not permitted.

[0014]

[Effects of the Invention]

As above explained, as access it does personal computer system which possesses the magnetic disk common device of this invention, from individual personal computer with all storage area of magnetic disk device of plural as storage area of magnetic disk device of 1 just, it makes that data of size which exceeds recording capacity of the magnetic disk device of 1 by at same time managing access privilege of each personal computer, is handled possible, Lumping together without it makes that magnetic disk device of plural is shared due to microcomputer of plural possible, keeping data with individual personal computer, there is an effect which it can manage.

[Brief Explanation of the Drawing (s)]

[Figure 1]

It is a one Working Example of personal computer system of this invention.

[Figure 2]

It is an example of partition \* control & table.

[Explanation of Symbols in Drawings]

- 1  
personal computer main body 1
- 10  
magnetic disk device
- 11  
magnetic disk device
- 12  
magnetic disk device
- 2  
personal computer main body 2
- 3

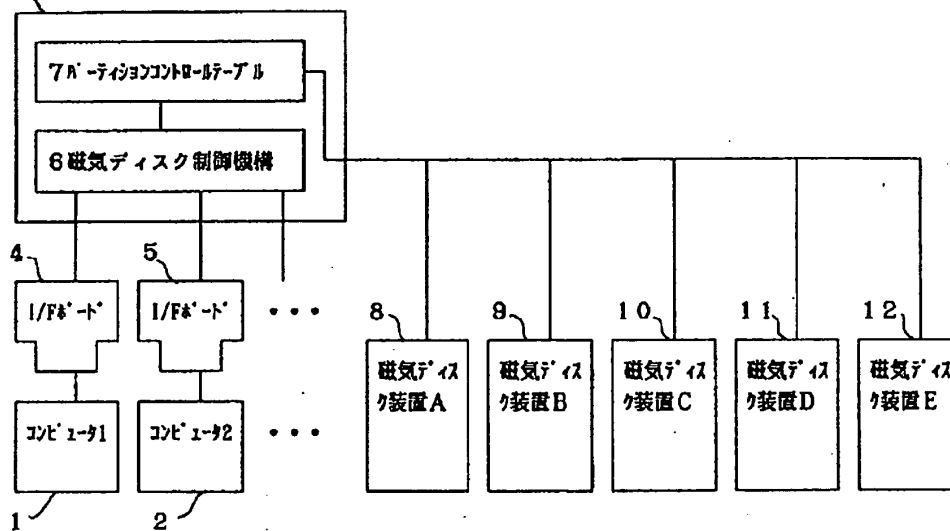
磁気ディスク共有装置	magnetic disk joint ownership device
4	4
インタフェースボード	interface board
5	5
インタフェースボード	interface board
6	6
磁気ディスク制御機構	magnetic disk control mechanism
7	7
パーティション・コントロール・テーブル	partition *Contro jp7 roll *table
8	8
磁気ディスク装置	magnetic disk device
9	9
磁気ディスク装置	magnetic disk device

Drawings

【図1】

[Figure 1]

3 磁気ディスク共有装置



【図2】

[Figure 2]

ノード名	コンピュータ名	アクセス権
ノード1	ノード1-タ1	RWCX
	ノード1-タ2	RWCX
	ノード1-タ3	RWCX
ノード2	ノード2-タ1	RW
	ノード2-タ3	R
ノード3	ノード3-タ1	R
	ノード3-タ2	R
ノードn	⋮	⋮

(R : 読みだし可 W : 書き込み可 C : 作成可 X : 実行可)

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **05181609 A**

(43) Date of publication of application: **23.07.93**

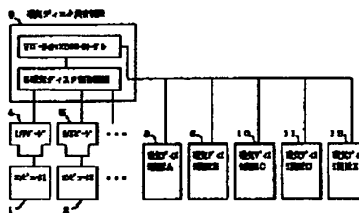
(51) Int. Cl. <b>G06F 3/06</b>	
(21) Application number: <b>04000333</b>	(71) Applicant: <b>NEC CORP</b>
(22) Date of filing: <b>08.01.92</b>	(72) Inventor: <b>HIRAI HIDEO</b>

(54) **PERSONAL COMPUTER SYSTEM**

(57) Abstract

PURPOSE: To provide a personal computer system by which plural magnetic disk devices can be shared with plural personal computers.

CONSTITUTION: A magnetic disk control mechanism 6 controls plural magnetic disk devices 8-12 as a single virtual magnetic disk device which uses all storage areas of these devices 8-12 as a storage area. A magnetic disk sharing device 3 contains the mechanism 6 and a partition control table 7 which controls the access right of each of personal computers 1, 2.... Thus each personal computer has an access to the virtual magnetic disk device based on each access right.



COPYRIGHT: (C)1993,JPO&Japio

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平5-181609

(43)公開日 平成5年(1993)7月23日

(51)Int.Cl.<sup>5</sup>

G 0 6 F 3/06

識別記号

3 0 1 Z 7165-5B

庁内整理番号

F I

技術表示箇所

審査請求 未請求 請求項の数1(全 4 頁)

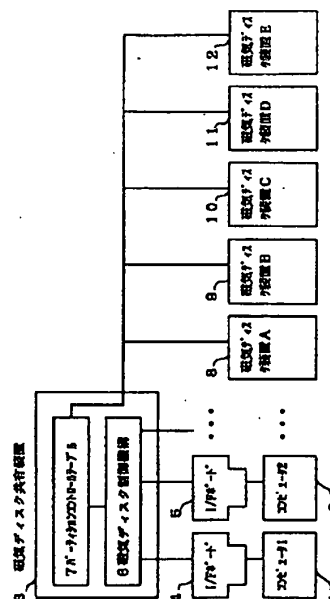
(21)出願番号	特願平4-333	(71)出願人	000004237 日本電気株式会社 東京都港区芝五丁目7番1号
(22)出願日	平成4年(1992)1月6日	(72)発明者	平井 秀生 東京都港区芝五丁目7番1号 日本電気株式会社内
		(74)代理人	弁理士 若林 忠

(54)【発明の名称】 パーソナルコンピュータシステム

(57)【要約】

【目的】 複数の磁気ディスク装置を複数のパーソナルコンピュータで共用できるパーソナルコンピュータシステムを提供する。

【構成】 複数の磁気ディスク装置8~12を、それらの全記憶領域を記憶領域とする1個の仮想磁気ディスク装置とみなして制御する磁気ディスク制御機構6と、仮想磁気ディスク装置の記憶領域のパーティションごとに指定する各パーソナルコンピュータ1, 2, ...のアクセス権を管理するパーティション・コントロール・テーブル7とを備えた磁気ディスク共有装置3を有し、各パーソナルコンピュータはそれぞれのアクセス権にしたがって仮想磁気ディスク装置にアクセスする。



## 【特許請求の範囲】

【請求項1】 複数のパーソナルコンピュータと複数の磁気ディスク装置を含むパーソナルコンピュータシステムにおいて、

前記複数の磁気ディスク装置を、該複数の磁気ディスク装置の全記憶領域をその記憶領域とする1個の仮想磁気ディスク装置とみなして制御する磁気ディスク装置の制御手段と、当該仮想磁気ディスク装置の記憶領域における前記複数のパーソナルコンピュータそれぞれの利用可能な権利を管理するセキュリティ管理手段とを備えた磁気ディスク共有装置を有し、

前記複数のパーソナルコンピュータはそれぞれ利用可能な前記権利にしたがって前記仮想磁気ディスク装置にアクセスすることを特徴とするパーソナルコンピュータシステム。

## 【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、複数のパーソナルコンピュータと複数の磁気ディスク装置を含むパーソナルコンピュータシステムに関する。

【0002】

【従来の技術】従来、パーソナルコンピュータシステムにおける磁気ディスク制御装置は1個の論理磁気ディスク装置に対して1個の物理磁気ディスク装置しか対応できず、上位ソフトウェア（オペレーティングシステム）の磁気ディスク管理も同様となっている。

【0003】また、磁気ディスク装置を複数のパーソナルコンピュータで共用できなかった。

【0004】

【発明が解決しようとする課題】上述した従来の磁気ディスク制御装置は、1個の論理磁気ディスク装置に1個の物理磁気ディスク装置しか対応できない仕様となっているため、1個の磁気ディスク装置の容量を超えるデータを扱うことができないという欠点があり、また磁気ディスク装置を複数のパーソナルコンピュータによって共用することができないという欠点があった。

【0005】本発明の目的は、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域として個々のパーソナルコンピュータからアクセスすることにより、一台の磁気ディスク装置の容量を超える大きさのデータを扱うことを可能にするともに、個々のパーソナルコンピュータのアクセス権を管理しつつ、複数のマイクロコンピュータによって複数の磁気ディスク装置を共用することのできる磁気ディスク制御装置を有するマイクロコンピュータシステムを提供することである。

【0006】

【課題を解決するための手段】本発明のパーソナルコンピュータシステムは、複数の磁気ディスク装置を、それら複数の磁気ディスク装置の全記憶領域をその記憶領域

とする1個の仮想磁気ディスク装置とみなして制御する磁気ディスク装置の制御手段と、その仮想磁気ディスク装置の記憶領域における複数のパーソナルコンピュータそれぞれの利用可能な権利を管理するセキュリティ管理手段とを備えた磁気ディスク共有装置を有し、複数のパーソナルコンピュータはそれぞれ利用可能な権利にしたがって仮想磁気ディスク装置にアクセスする。

【0007】

【作用】個々のパーソナルコンピュータにとり、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域であるかのようにアクセスすることを可能とし、その1個の仮想磁気ディスク装置の記憶領域に対する複数のパーソナルコンピュータそれぞれの利用可能な権利は個々のマイクロコンピュータごとに指定され管理される。

【0008】

【実施例】次に、本発明の実施例について図面を参照して説明する。

【0009】図1は本発明の、磁気ディスク共有装置を含むパーソナルコンピュータシステムの一実施例の構成を示すブロック図である。

【0010】図1において、パーソナルコンピュータ本体1、2、・・・はインタフェースボード4、5、・・・を介して磁気ディスク共有装置3に接続されている。また、磁気ディスク共有装置3には磁気ディスク装置8～12が接続されている。磁気ディスク共有装置3は、磁気ディスク制御機構6とパーティション・コントロール・テーブル7から構成されている。

【0011】パーソナルコンピュータ1、2、・・・から磁気ディスク装置8～12へのアクセス要求は、磁気ディスク用インタフェースボード4、5、・・・を通じ磁気ディスク制御機構6に通知され、磁気ディスク制御機構6において磁気ディスク装置8～12にまたがる仮想的な磁気ディスク装置に対するアクセス要求に変換される。以上の処理によりパーソナルコンピュータ本体からは、磁気ディスク装置8～12を、磁気ディスク装置8～12の全記憶領域を自らの記録領域とする仮想化された一つの磁気ディスク装置として扱うことが可能となる。

【0012】セキュリティ管理は、上述の仮想磁気ディスク装置の記憶領域を区別し、区別された各部分（パーティションと云う）に、個々のパーソナルコンピュータごとの利用可能な権利を設定し、不正なアクセスを防ぐためのもので、パーティション・コントロール・テーブル7を作成して行われる。パーティションへのアクセス権には、R（読み出し）、W（書き込み）、C（作成）、D（消去）、X（実行）がある。

【0013】図2はパーティション・コントロール・テーブルの例である。パーソナルコンピュータ1は、パーティション1に対し読み出し、書き込み、作成、実行が

可能であり、パーティション2に対し読み出し、書き込みが可能であり、パーティション3に対し読み出しが可能である。パーソナルコンピュータ2は、パーティション1に対し読み出し、書き込み、作成、実行が可能であり、パーティション3に対し読み出しが可能である。パーソナルコンピュータ3は、パーティション1に対し読み出し、書き込み、作成、実行が可能であり、パーティション2に対し読み出しが可能である。上記セキュリティ管理手段により、パーソナルコンピュータからの利用の許されていない不正なアクセスを防止することが可能となる。

【0014】

【発明の効果】以上説明したように、本発明の磁気ディスク共用装置を有するパーソナルコンピュータシステムは、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域として個々のパーソナルコンピュータからアクセスし、かつ各パーソナルコンピュータのアクセス権を管理することにより、1個の磁気ディスク装置の記憶容量を超える大きさのデータを扱

＊うことを可能にするとともに、複数のマイクロコンピュータにより複数の磁気ディスク装置を共用することを可能にし、データを個々のパーソナルコンピュータで保管することなく一括して管理することができる効果がある。

【図面の簡単な説明】

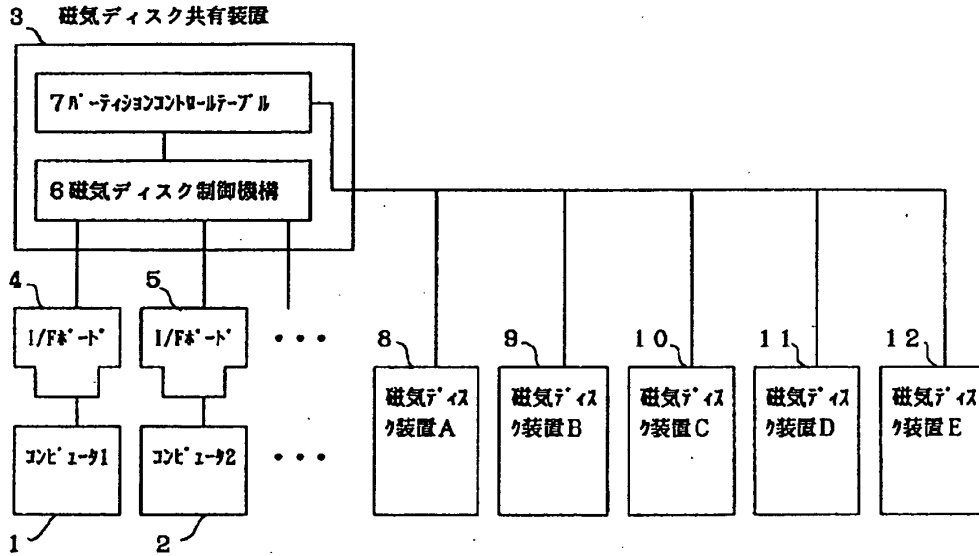
【図1】本発明のパーソナルコンピュータシステムの一実施例である。

【図2】パーティション・コントロール・テーブルの例である。

【符号の説明】

- 1 パーソナルコンピュータ本体1
- 2 パーソナルコンピュータ本体2
- 3 磁気ディスク共用装置
- 4～5 インタフェースボード
- 6 磁気ディスク制御機構
- 7 パーティション・コントロール・テーブル
- 8～12 磁気ディスク装置

【図1】





【図2】

ノード名	コンピュータ名	アクセス権
ノード1	ノード1-1	RWCX
	ノード1-2	RWCX
	ノード1-3	RWCX
ノード2	ノード2-1	RW
	ノード2-3	R
ノード3	ノード3-1	R
	ノード3-2	R
ノードn	⋮	⋮

(R：読みだし可 W：書き込み可 C：作成可 X：実行可)

JP1997185594A

1997-7-15

~~B/A~~  
B21

**Bibliographic Fields**

**Document Identity**

(19)【発行国】

日本国特許庁(JP)

(12)【公報種別】

公開特許公報(A)

(11)【公開番号】

特開平9-185594

(43)【公開日】

平成9年(1997)7月15日

**Public Availability**

(43)【公開日】

平成9年(1997)7月15日

**Technical**

(54)【発明の名称】

直接バルク・データ転送

(51)【国際特許分類第6版】

G06F 15/163

【FI】

G06F 15/16 320 R

320 V

【請求項の数】

6

【出願形態】

OL

【全頁数】

25

**Filing**

【審査請求】

未請求

(21)【出願番号】

特願平8-301770

(22)【出願日】

平成8年(1996)11月13日

(19) [Publication Office]

Japan Patent Office (JP)

(12) [Kind of Document]

Unexamined Patent Publication (A)

(11) [Publication Number of Unexamined Application]

Japan Unexamined Patent Publication Hei 9- 185594

(43) [Publication Date of Unexamined Application]

1997 (1997) July 15\*

(43) [Publication Date of Unexamined Application]

1997 (1997) July 15\*

(54) [Title of Invention]

**DIRECTLY BULK \*DATA TRANSFER**

(51) [International Patent Classification, 6th Edition]

G06F15/163

[FI]

G06F15/16320R

320V

[Number of Claims]

6

[Form of Application]

OL

[Number of Pages in Document]

25

[Request for Examination]

Unrequested

(21) [Application Number]

Japan Patent Application Hei 8- 301770

(22) [Application Date]

1996 (1996) November 13\*

JP1997185594A

1997-7-15

**Foreign Priority**

(31)【優先権主張番号】

08/556618

(32)【優先日】

1995年11月13日

(33)【優先権主張国】

米国(US)

(31) [Priority Application Number]

08/556618

(32) [Priority Date]

1995 November 13\*

(33) [Priority Country]

United States (U.S. Patent )

**Parties**

**Applicants**

(71)【出願人】

【識別番号】

391058071

【氏名又は名称】

タンデム コンピューターズ インコーポレイテッド

【氏名又は名称原語表記】

TANDEM COMPUTERS INCORPORATED

【住所又は居所】

アメリカ合衆国 カリフォルニア州 95014 クーパーティノ ノース タンタウ アベニュー 10435

(71) [Applicant]

[Identification Number]

391058071

[Name]

TANDEM COMPUTERS INCORPORATED

[Name in Original Language]

TAN DEM COMPUTERS Incorporated

[Address]

United States of America California 95014Cupertino north [tantau] Avenue 10435

**Inventors**

(72)【発明者】

【氏名】

トッド ダブリュー スプレングル

【住所又は居所】

アメリカ合衆国 カリフォルニア州 94086 サニーヴェイル ヴァスケズ アベニュー 387

(72)【発明者】

【氏名】

スリニヴァサ ディー マーシー

【住所又は居所】

アメリカ合衆国 カリフォルニア州 95131 サンホセ ゴールデンレイク ロード 1410

(72)【発明者】

【氏名】

アニル カートリ

(72) [Inventor]

[Name]

[toddo] W [supurenkuru]

[Address]

United States of America California 94086Sunnyvale [vuasukezu] Avenue 387

(72) [Inventor]

[Name]

[surinivusa] D. -mer C.

[Address]

United States of America California 95131San Jose golden lake load 1410

(72) [Inventor]

[Name]

anil cart jp9

JP1997185594A

1997-7-15

【住所又は居所】

アメリカ合衆国 カリフォルニア州 95148 サン  
ホセ ヘリテージ エステータス ドライヴ 328  
1

[Address]

United States of America California 95148 San Jose [heriteejij]  
[esuteetasu] drive 3281

Agents

(74)【代理人】

【弁理士】

【氏名又は名称】

中村 稔 (外6名)

(74) [Attorney(s) Representing All Applicants]

[Patent Attorney]

[Name]

Nakamura Minoru (6 others)

Abstract

(57)【要約】

(修正有)

【課題】

CPUsと記憶ディスクとの間のデータの転送中の  
類似するデータ・コピーを除去する。

【解決手段】

記憶処理 130 は、記憶装置 100~105 及び 110~  
115 を有する 30 へのアクセスを制御する。

ソフトウェア・ルーチンは、要求 CPU22 による記  
憶装置 30 への直接アクセスを供給するために  
用いられる。

要求 CPU22 のバッファ 160 に対する仮想メモリ・  
アドレスは、要求 CPU22 で生成される。

記憶装置アクセス要求と共に仮想メモリ・アドレ  
スは、記憶処理 130 を含んでいる CPU20 に送ら  
れる。

ワーク要求は、記憶処理 130 から記憶装置 30  
へ送られる仮想メモリ・アドレスを含む。

次いで、データは、要求 CPU22 と記憶装置 30 と  
の間で直接転送される。

記憶装置 30 は、次いでワーク要求に応答する。

(57) [Abstract]

(There is an amendment.)

[Problems to be Solved by the Invention]

It is in midst of transferring data between CPUs and the  
storage disc and data \*copy which resembles is removed.

[Means to Solve the Problems]

recording routine 130 controls access to 30 where it possesses  
storage device 100~105 and 110 - 115.

software \*routine is used in order to supply directly access to  
storage device 30 with required CPU 22.

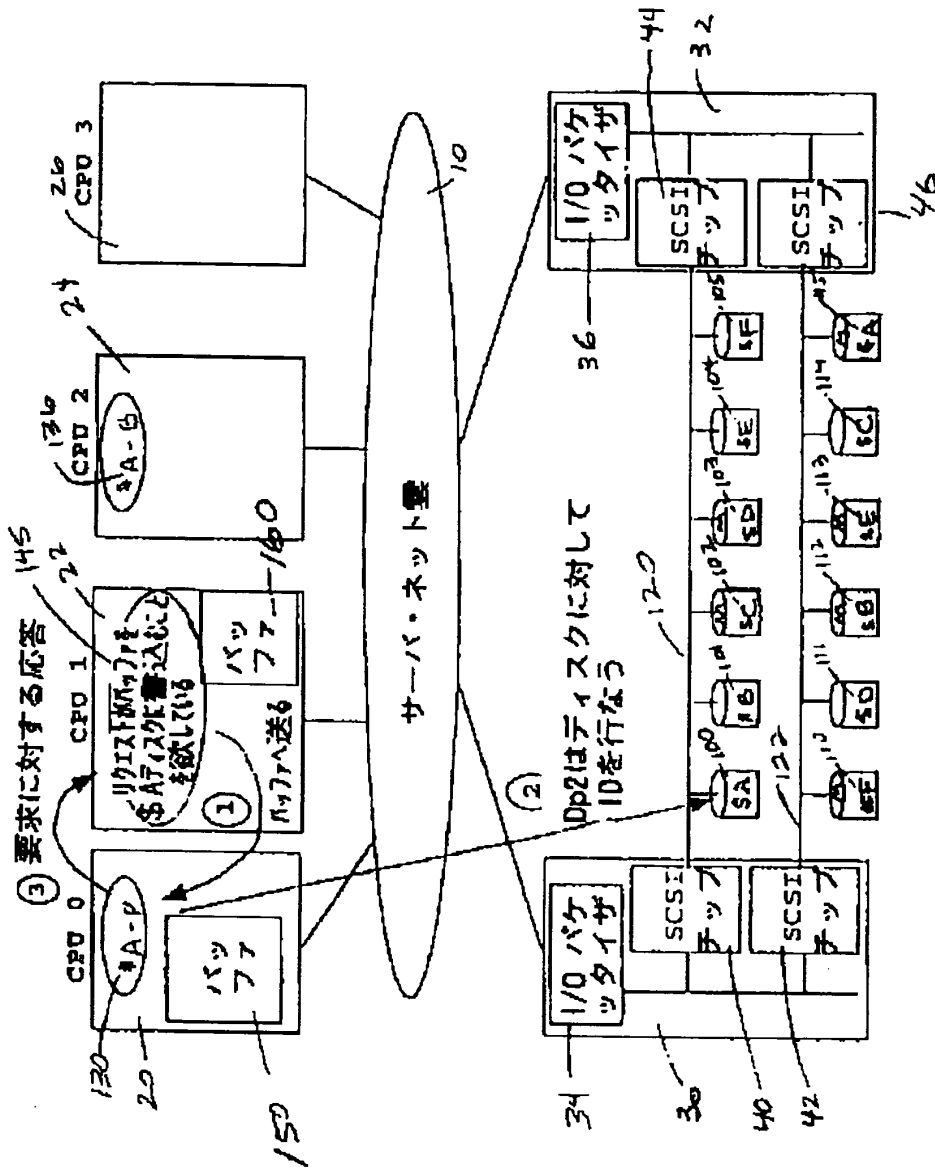
Hypothetical memory \*address for buffer 160 of required  
CPU 22 is formed with required CPU 22.

With storage device access request hypothetical memory  
\*address is sent to CPU 20 which includes recording routine  
130.

work request includes hypothetical memory \*address which is  
sent to the storage device 30 from recording routine 130.

Next, data is transferred directly required CPU 22 and  
between storage device 30.

As for storage device 30, you respond to work request next.



Claims

【特許請求の範囲】

[Claim (s)]

【請求項 1】

[Claim 1]

データを転送するデータ処理システムであって、要求 CPU を含んでいる複数の中央処理装置(CPU);前記 CPU の一つが記憶装置へのアクセスを制御する少なくとも一つの記憶装置;前記複数の CPU と前記記憶装置とを相互接続するネットワーク;前記要求 CPU による前記記憶装置の直接アクセスを供給し、前記要求 CPU のバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に前記仮想メモリ・アドレスを、前記記憶装置へのアクセスを制御している前記 CPU の前記一つに供給する前記要求 CPU における手段、前記ワーク要求に回答しておりかつ前記ネットワークを通して前記 CPU の前記一つと直接インターフェイスしている前記記憶装置に前記仮想メモリ・アドレスを含んでいるワーク要求を送る前記 CPU の前記一つにおける手段、を含んでいるアクセス手段;を備え、

前記データは、前記要求 CPU と前記記憶装置との間で直接転送されることを特徴とするデータ処理システム。

【請求項 2】

前記直接アクセスは、前記要求 CPU に前記記憶装置から前記仮想メモリ・アドレスにおける前記バッファ・メモリの中にデータを読取らせることを特徴とする請求項 1 に記載のデータ処理システム。

【請求項 3】

前記データは、それぞれが、宛先ノード識別、ソース・ノード識別、前記仮想メモリ・アドレス、及び複数のデータ・ワードを含んでいるデータ・パケットで送信されることを特徴とする請求項 1 に記載のデータ処理システム。

【請求項 4】

前記記憶装置は、転送のためのデータを含んでいる前記データ・パケットのそれぞれが前記要求 CPU に送信された後で前記 CPU の前記一つにアドバイスをする手段を含むことを特徴とする請求項 3 に記載のデータ処理システム。

【請求項 5】

With data processing system which transfers data , central processing unit of plural which includes required CPU (CPU); Aforementioned storage device to supply access directly with CPUs of storage device ; aforementioned plural of at least one where one of theaforementioned CPUs controls access to storage device and network ; aforementioned required CPU which interconnect does aforementioned storage device , hypothetical memory \*address for buffer of aforementionedrequired CPU only formation with storage device access request aforementionedhypothetical memory \*address , We to have responded to means. aforementioned work request in theaforementioned required CPU which is supplied to aforementioned one of aforementioned CPUs which controls access to theaforementioned storage device and through aforementioned network theaforementioned one of aforementioned CPUs directly access means ; which includes means. in aforementioned one of theaforementioned CPUs which sends work request which includes theaforementioned hypothetical memory \*address in aforementioned storage device which interface has been done having,

As for aforementioned data , data processing system . which designates theaforementioned required CPU and that it is directly transferredbetween aforementioned storage device as feature

[Claim 2 ]

Description above directly access data processing system . which is stated in the Claim 1 which designates that data is made to grasp in theaforementioned buffer \*memory in aforementioned required CPU from theaforementioned storage device in aforementioned hypothetical memory \*address asfeature

[Claim 3 ]

As for aforementioned data , each one, data processing system . which is statedin Claim 1 which designates that it is transmitted with data \*packet which includes addressee node identification, source \*node identification, theaforementioned hypothetical memory \*address , and data \*word of plural asfeature

[Claim 4 ]

As for aforementioned storage device , after each one of aforementioned data \*packet which includes data for transferring was transmitted to theaforementioned required CPU data processing system . which is stated in Claim 3 which designates that means which does advise in theaforementioned one of aforementioned CPUs is included asfeature

[Claim 5 ]

前記直接アクセスは、前記要求 CPU に、前記データの転送のために前記仮想メモリ・アドレスにおける前記バッファをアクセスしている前記記憶装置の中に書込ませることを特徴とする請求項 1 に記載のデータ処理システム。

## 【請求項 6】

前記アクセス手段は、前記要求 CPU の前記バッファに対する前記仮想メモリ・アドレスを生成しかつ前記記憶装置アクセス要求と共に前記仮想メモリ・アドレスを前記 CPU の前記一つに供給する少なくとも一つのソフトウェア・ルーチンを含むことを特徴とする請求項 1 に記載のデータ処理システム。

## Specification

## 【発明の詳細な説明】

【0001】

## 【産業上の利用分野】

本発明は、一般にデータ処理システムにおける直接データ転送に関し、より特定のにはプロセッサ及び入力/出力通信に対する接続性を供給するネットワークを介してシステムで行われる直接バルク入力/出力転送に関する。

【0002】

## 【従来の技術】

サード・パーティ転送システムは、データ転送に対する種々のスキームを供給する。

例えば、高性能記憶システム(High-Performance Storage System (HPSS))は、高集合体 I/O 処理能力(スループット)を達成するためにネットワークにわたり同時入力/出力(I/O)オペレーションを統合(調整)することができる高度な、分散型階層記憶管理システムである。

HPSS は、国立記憶研究所(National Storage Laboratory) で開発中の次世代ソフトウェア・システムである。

HPSSn についての更なる情報は、次に示すインターネット上のワールド・ワイド・ウェブ・ページを通して取得することができる:

[http://www.ccs.ornl.gov/HPSS/HPSS \\_\\_overview.html](http://www.ccs.ornl.gov/HPSS/HPSS__overview.html)

[http://www.ccs.ornl.gov/HPSS/HPSS\\_overview.html](http://www.ccs.ornl.gov/HPSS/HPSS_overview.html)

Description above directly access in aforementioned required CPU , in aforementioned storage device which access does the aforementioned buffer in aforementioned hypothetical memory \*address for transferring aforementioned data entry data processing system . which is stated in Claim 1 which designates that it can increase as feature

## [Claim 6 ]

As for aforementioned access means , aforementioned hypothetical memory \*address for aforementioned buffer of aforementioned required CPU only formation with aforementioned storage device access request data processing system . which is stated in Claim 1 which designates that software \*routine of at least one which supplies aforementioned hypothetical memory \*address to aforementioned one of aforementioned CPUs is included as feature

## [Description of the Invention ]

【0001】

## [Field of Industrial Application ]

As for this invention, it regards data transfer in data processing system generally directly, from specifically through network which supplies connectivity for processor and input/output communication , it regards bulk input/output which is done with the system transfer directly.

【0002】

## [Prior Art ]

third \*per T. transfer system supplies various scheme for data transfer .

High-Performance storage system (HPSS) is high-level , distributed type hierarchy memory management system it can integrate (Adjustment) simultaneous input/output (I/O ) operation in order to achieve high assembly I/O capacity (throughput ) over network . for example high performance storage system

HPSS is next generation software \*system which is in midst of developing with the national storage research laboratory (National storage Laboratory ).

You can acquire further data concerning HPSSn, through the world \*wide \*web page on Internet which is shown next:

http://www.llnl.gov/liv	__comp/nsl/hpss/hpss.html
http://www.llnl.gov/liv	*comp/ns l/hr ps s/h ps s.html

からシンク・デバイス(sink device) までデータを転送するために用いられるムーバ(Mover) を開示する。

このムーバは、また、一組のデバイス制御オペレーションを実行する。

【0003】

ファイバ・チャネル(FC)は、また、サード・パーティ転送を供給する。

FC は、ワークステーション、メインフレーム、スーパーコンピュータ、デスクトップ・コンピュータ、記憶装置、ディスプレイ及び他の周辺装置間でデータの転送を許容する。

FC についての更なる情報は、次に示すインターネット上のワールド・ワイド・ウェブ・ページを通して取得することができる:

<http://www.amdahl.com/ext/CARP/FCA/FCA.html>

そして、IEEE 記憶システム標準研究グループは、オープン記憶システム相互接続(Open Storage Systems Interconnection(OSSI)) に対するドラフト基準モデル(draft Reference Model) を生成した。

次に示すのは、ドラフト OSSI ドキュメントのバージョン 5 からのものである: "2.3.3 制御及びデータ・フローの分離

"OSSI モデルは、クライアント、データ・ソース、及びデータ・シンクの間で発生しているデータ・フローから制御フローを区別する。

制御フローは、クライアントとデータ・ソースまたはデータ・シンクとの間で要求、応答、及び非同期通知をキャリアーする。

データ・ソースとデータ・シンクとの間の制御フローは、データのフローを管理するためにソース・シンク・プロトコル情報をキャリアーする。

データ・フローは、ソースからシンクまでのみパスする。

制御及びデータ・フローを論理的に分離することによって、OSSI モデルは、個別のインプリメンテーションを通る各フローを最適化することの可能性を供給する。

"

[muuba ] (Mover ) which is used in order to transfer data to empty sink \*device (sinkdevice ) is disclosed.

As for this [muuba ], in addition, device control operation of one set is executed.

【0003】

As for fiber \*channel (FC ), in addition, third \*per T. transfer is supplied.

FC allows transfer of data between workstation , mainframe , server computer , desktop \*computer , storage device , display and theother peripheral .

You can acquire further data concerning FC, through the world \*wide \*web page on Internet which is shown next:

<http://www.amdahl.com/ext/CARP/FCA/FCA.html>

draft reference model (draftReferenceModel ) for Openstorage Systems interconnection (OSSI ) was formed. And, as for IEEE storage system standard research group , open storage system interconnect

Fact that it shows next is something from version 5 of the draft OSSI document ,;" 2.3.3 controls and separation of data \*flow

"OSSI model distinguishes control flow from client , data \*source , and data \*flow which occurs between data \*sink .

client and it requires control flow , data \*source or data \*sink between, responds, and it notifies asynchronous carry .

control flow between data \*source and data \*sink carry does source -sink \*protocol data inorder to manage flow of data .

To only sink pass it does data \*flow , from source .

By fact that control and data \*flow are separated into logical ,OSSI model supplies possibility of thing which optimization does each flow which passes by individual [inpurimenteeshon ].

"



## "2.3.4 サード・パーティ転送

"OSSI モデルは、サード・パーティの制御下で、データを独立のソース及びシンク間で直接フローさせ、エージェントまたはクライアントを起動し、かつ制御する。

各エンティティは、データ・フロー制御、エラー報告、または転送を開始し、かつ終了することのようなオペレーションを個別に実行する。

"

OSSI についての更なる情報は、次のワールド・ワイド・ウェブ・ページを介して取得することができる:

<http://www.arl.mil/IEEE/ssswg.html>

全ての目的のためにここに参照文献として採り入れた、1995年6月7日に出版された米国特許出願第 08/486,217 号(添付資料 A として添付される)は、プロセッサ及び入力/出力通信間の接続性または結合性を供給する高信頼システム・エリア・ネットワークにおける多重処理システムに対する必要性に応ずる。

この特許出願(以下、添付特許出願と呼ぶ)は、フェール・ファースト、フェール・ファンクショナル、フォールト・トレラント・マイクロプロセッサ・システムを供給する。

添付特許出願に開示されたアーキテクチャは、あらゆる中央処理装置(CPU)を入力/出力(I/O)コントローラ(サーバ・ネット・アダプタとも呼ばれる)と通信させるサーバ・ネットワーク雲(server network cloud)を含む。

従って、I/O コントローラは、CPU によってアドレス指定されることができ、かつ CPU は、I/O コントローラによってアドレス指定されることができ、

このサーバ・ネットワーク雲は、添付特許出願ではサーバ・ネットワークと呼ばれている。

本願と共に添付特許出願は、タンデム・コンピュータ・インコーポレーティッドに譲渡されている。

[0004]

広く言えば、添付特許出願に開示された発明は、多重サブ処理システムを備えた処理システムを含む。

各サブ処理システムは、主処理構成要素として、CPU を有する。

この CPU は、同時に命令ストリームからの各命令を実行するためにロック・ステップ同期(lock-step)

## " 2.3.4 third \*per T. transfers

"OSSI model, under controlling third \*per T., flow doing data directly between independent source and sink, only starting controls the agent or client.

Each entity, data \*flow control, error report or transfer only start executes operation like thing which ends individually.

"

Through following world \*wide \*web page, you can acquire further data concerning OSSI:

<http://www.arl.mil/IEEE/ssswg.html>

As for U.S. Patent Application No. 08/486, 217 number (It is attached as attachment material A) which it adopted here because of all objective is applied in 1995 June 7 days as cited reference, it responds to necessity for multiple processing system in high reliability system \*area \*network which supplies connectivity or bonding ability between processor and input/output communication.

this patent application (Below, it calls attachment patent application) [feeru] \* fast, [feeru] \* functional, [fooruto] \* supplies pick-up run jp7 \*microprocessor \*system.

architecture which is disclosed in attachment patent application includes the server \*network cloud (servernetcloud) which input/output (I/O) controller (Also server \*network \*adapter is called) with communication does all central processing unit (CPU).

Therefore, as for I/O controller, address it is possible with CPU to be appointed, at same time address with I/O controller to be appointed it is possible CPU.

this server \*network cloud with attachment patent application is called server \*network.

With this application attachment patent application transfer is made tandem \*computer \*incorporated.

[0004]

Speaking widely, invention which is disclosed in attachment patent application includes processing system which has multiple sub processing system.

Each sub processing system has CPU as main treatment component.

this CPU includes processor of pair which in order to execute each command from command stream simultaneously

ep synchronized)方式で動作する一対のプロセッサを含む。

サブ処理システムのそれぞれは、より大きな処理システムの種々のコンポーネント間に冗長通信経路を供給する I/O システム・エリア・ネットワークを更に含む。

これらの種々のコンポーネントは、CPU 及びアサートされた周辺デバイス(例えば、マス記憶装置、プリンタ、等)を含む。

これらの冗長通信経路は、また、より大きな総括処理システムを形成するサブ-プロセッサ間に存在することができる。

処理システムのコンポーネント間(例えば、第 1 の CPU と第 2 の CPU の間、または CPU と周辺デバイスとの間)の通信は、パケットに含まれるメッセージを形成しかつ送信することによってインプリメントされる。

好ましい実施例では、各パケットは、64 バイトのデータを含む。

これらのパケットは、システム・エリア・ネットワーク構造(サーバ・ネットワークと呼ぶ)により送信またはソース・コンポーネント(例えば、CPU)から宛先構成要素(例えば、周辺デバイス)へ送られる。

[0005]

このシステム・エリア・ネットワーク構造は、複数の相互接続リンクによって相互接続される多数のルータ構成要素を含む。

パケットのソースからパケットの宛先へパケットを送る。添付特許出願に開示されたルータは、それら自身パケットを発生しない。

ルータは、一つのリンク上の入力パケットを取り入れてかつそれをその宛先に対して適切なリンク上に送り出すことによってパケット・スイッチとして動作する。

ルータ構成要素は、処理システムの送信コンポーネントから宛先コンポーネントまでの適切なまたは利用可能な通信経路を選択する責任がある。

通信経路は、メッセージ・パケットに含まれる情報に基づく。

それゆえに、ルータ構成要素のルーティング能力(ケイパビリティ)は、周辺デバイスへの通信経路を有する CPU の I/O システムを供給する。

[0006]

添付特許出願に開示されたアーキテクチャは、

operates with lock \*step synchronization (lock-steps ynchronized ) system .

Each one of sub processing system furthermore includes input/output system \*area \*network which supplies redundant communication path between various component of a larger processing system .

These various component CPU and include peripheral device (for example mass storage device , printer , etc) which assert isdone .

As for these redundant communication path , in addition , it can exist between sub -processor which forms a larger general processing system .

communication of (Between between, or CPU and peripheral device for example first CPU and second CPU ) between component of processing system only formation the implementation is done message which is included in packet by factthat it transmits .

With desirable Working Example , as for each packet , data of 64 byte isincluded .

These packet are sent to addressee component (for example peripheral device ) from transmission or source \*component (for example CPU) by system \*area \*network structure (It calls server \*network ) .

[0005]

this system \*area \*network structure includes multiple router component which interconnect is done with interconnect link of plural .

router which from source of packet sends packet to the addressee of packet , is disclosed in attachment patent application does notgenerate those itself packet .

router , adopting input packet on link of one , andoperates by fact that it sends out that on appropriate link vis-a-vis the addressee as packet \*Switch .

router component is a responsibility which selects appropriate or practical communication path to addressee component from transmission component of processing system .

communication path is based on data which is included in message \*packet .

Consequently, routing capacity ( [keipabirity ] ) of router component supplies input/output system of the CPUs which possesses communication path to peripheral device .

[0006]

As for architecture which is disclosed in attachment patent

各ディスクを管理するためにディスク処理ペアを用い、ディスク処理ペアの半分は、1次ディスク処理でありかつ他の半分は、バックアップ・ディスク処理である。

更に、SCSI チェイン上のディスクを制御しているディスク処理は、二つの CPUs に制限されず、かつディスク処理は、複数の CPUs 中で走るように構成することができる。

添付特許出願のサーバ・ネット雲が用いられるときには、CPUs 及びサーバ・ネット・アダプタの両方は、CPU メモリに対する読み取り及び書き込みサーバ・ネット雲トランザクションを発生することができる。

図 1 は、記憶装置のアーキテクチャの一例を示す。

この構成は、添付特許出願に開示されたサーバ・ネット雲 10 を含む。

ディスク/記憶コントローラ 30 及び 32 と共に CPUs 20, 22, 24 及び 26 は、サーバ・ネット雲 10 に接続される。

サーバ・ネット・アダプタ 30 及び 32 は、SCSI チップ 40, 42, 44 及び 46 と共に I/O パケッタイザ (packetizers) 34 及び 36 を含む。

I/O パケッタイザは、ネットワーク・プロトコルからのデータ・パケットをバス・プロトコルに変換する。

この構成は、また、(合計 12 の記憶ディスク 100~105 及び 110~115 に対して)二つの SCSI チェイン 120 及び 122 をハング・オフしている (hanging off) 6 つの SCSI ディスク・ペア 100~105 及び 110~115 を示す。

ディスクは、1次ディスク 100(\$A), 101(\$B), 104(\$E), 105(\$F), 111(\$D) 及び 114(\$C) 及びミラー・ディスク 102(\$C), 103(\$D), 110(\$F), 112(\$B), 113(\$E) 及び 115(\$A) として構成される。

4 つの CPUs (CPU 20, CPU 22, CPU 24 及び CPU 26) は、6 つのディスク・ペア 100~105 及び 110~115 を制御するディスク処理を収容する。

1次ディスク処理 (\$A-P, \$B-P, \$C-P, \$D-P, \$E-P 及び \$F-P) 及びバックアップ・ディスク処理 (\$A-A, \$B-B, \$C-B, \$D-B, \$E-B 及び \$F-B) は、4 つの CPUs 20, 22, 24 及び 26 の中に散乱される。

例えば、記憶ディスク 100(\$A) に対する 1次ディスク処理 130(\$A-P) は、CPU 20 に配置されるし、かつ記憶ディスク 102(\$C ミラー) に対するバックアップ・ディスク処理 133(\$C-P) は、CPU 1

application, in order to manage each disk as for half of disk treatment pair, and as for other half, it is a backup \*disk treatment in primary disk treatment making use of disc treatment pair.

Furthermore, disk treatment which controls disk on SCSI chain is not restricted by CPUs of two, at same time disk treatment, in order to run in CPUs of plural, configuration is possible.

When using server \*network cloud of attachment patent application, both of the CPUs and server \*network \*adapter can generate reading and entry server \*network cloud transaction for CPU memory.

Figure 1 shows one example of architecture of storage device.

This configuration includes server \*network cloud 10 which is disclosed in the attachment patent application.

With disk /storage controller 30 and 32 CPUs 20, 22, 24 and 26 are connected to server \*network cloud 10.

server \*network \*adapter 30 and 32 with SCSI tip 40, 42, 44 and 46 I/O [pakettaiza ] (packetizers ) include 34 and 36.

I/O [pakettaiza ] converts data \*packet from network \*protocol to bus \*protocol.

As for this configuration, in addition, SCSI chain 120 of (storage disc 100~105 of total 12 and in 110 - 115 confronting ) two and 122 is done [hangu ] \* off, (hanging off) 6 SCSI disk \*pair 100~105 and 110 - 115 is shown.

disk configuration is done primary disk 100 (\$A ), 101 (\$B ), 104 (\$E ), 105 (\$F ), 111 (\$D ) and 114 (\$C ) and mirror \*disk 102 (\$C ), 103 (\$D ), 110 (\$F ), 112 (\$B ), 113 (\$E ) and 115 (\$A ) as.

CPUs (CPU 020, CPU 122: CPU 224 and CPU 326 ) of 4 accommodates 6 disk \*pair 100~105 and disk treatment which controls 110 - 115.

primary disk treatment (\$A-P, \$B-P, \$C-P, \$D-P, \$E-P and \$F-P ) and backup \*disk treatment (\$A-A, \$B-B, \$C-B, \$D-B, \$E-B and \$F-B ) scattering makes the CPUs 20, 22, 24 of 4 and in 26.

primary disk treatment 130 (\$A-P ) for for example storage disc 100 (\$A ) is arranged in CPU 020 and lacquer, backup \*disk Process 133 (\$C-P ) at same time for storage disc 102 (\$C mirror ) can be arranged in CPU 122.

22 に配置される。

ディスク処理 130-141 は、二つの CPUs 以上に (図 1 に示すように) 配置することができる。

別の構成では、8 つの記憶ディスク・ペアは、(合計 16 の記憶ディスクに対して) 二つの SCSI チェインをハング・オフすることができ、かつエキストラ(余分な)SCSI チップは、外部記憶デバイスに対する各サーバ・ネット・アダプタに配置することができる。

[0007]

添付特許出願に示したように、ディスク 100 に書込むための CPU1 22 に配置される要求処理 145 に対して、要求処理 145 は、書込みデータ・メッセージを CPU0 20 に配置されたディスク処理 130 にまず送る(ディスク処理 130 は、ディスク 100 を制御する)。

次いで、ディスク処理 130 は、データにわたりチェックサムを計算する。

チェックサムは、データのブロックに対するデータ安全性が転送されることを確実にする。

次いで データのブロックは、サーバ・ネット雲 10 を通して CPU0 20 からディスク 100 に転送される。

この転送が終了したときに、ディスク処理 130 は、要求処理 145 に応答する。

[0008]

【発明が解決しようとする課題】

図 2 は、ディスク転送の一例を示す。

図 2 は、図 1 に含まれる同じ構成要素のほとんどを有する。

これらの構成要素に加えて、CPU0 20 に配置されるバッファ 150 及び CPU1 22 に配置されるバッファ 160 が示されている。

この例では、要求処理及びディスク処理は、異なる CPUs に配置される。

これらの処理は、また、同じ CPU に配置されることもできる。

段階 1 では、要求処理は、バッファ 160 から \$A ディスク 100 にデータを書込むことを欲する。

従って、CPU0 20 に配置された、ディスク処理 130 が \$A ディスク 100 に対するディスク処理なので、バッファ 160 に配置されたデータは、CPU0 20 に配置されたバッファ 150 に送られる。

( \$Cmirror ) can be arranged in CPU 122.

disk treatment 130 - 141 (It shows in Figure 1 way ) can be arranged in CPU s or greater of the two .

With another configuration , as for 8 horn storage disc \*pair , SCSI chain of (In storage disc of total 16 confronting ) two [hangu ]\* off it is possible , at same time it can arrange extra (excess ) SCSI tip , to do , in each server \*network \*adapter for outside storage device .

[0007]

As shown in attachment patent application , as for request treatment 145 , the entry data \*message is sent to disk treatment 130 which is arranged in CPU 020 first , entry \* vis-a-vis request treatment 145 which is arranged in CPU 122 of for sake of in disk 100 , (disk treatment 130 controls disk 100 ) .

Next , as for disk treatment 130 , checksum is calculated over the data .

checksum makes that data integrity for block of data is transferred secure .

Next block of data from CPU 020 is transferred to disk 100 through server \*network cloud 10 .

When this transfer ends , as for disk treatment 130 , you respond to request treatment 145 .

[0008]

[Problems to be Solved by the Invention ]

Figure 2 shows one example of disk transfer .

Figure 2 has majority of same component which are included in the Figure 1 .

In addition to these component , buffer 150 which is arranged in CPU 020 and buffer 160 which is arranged in CPU 122 are shown .

With this example , as for request treatment and disk treatment , it is arranged in different CPUs .

As for these treatments , in addition , it is possible also to be arranged in same CPU .

With step 1 , as for request treatment , from buffer 160 data entry \* thing is desired in \$A disk 100 .

Therefore , it was arranged in CPU 020 , because it is a disk treatment disk treatment 130 for \$A disk 100 , data which is arranged in buffer 160 is sent to buffer 150 which is arranged in the CPU 020 .

JP1997185594A

1997-7-15

段階 2 では、ディスク処理 130 は、バッファ 150 に配置されたデータをディスク 100 に書込む。

段階 3 では、ディスク 100 へのデータの転送が終了したので、ディスク処理 130 は、要求処理 145 に応答する。

[0009]

この構成では、データは、要求処理を有する CPU から関連ディスク処理を有する中間 CPU にコピーされ、そしてその中間 CPU からディスクにコピーされる。

CPUs と記憶ディスクとの間のデータの転送の間、類似するデータ・コピーを除去することが望ましい。

本発明の目的は、上記従来技術の問題に鑑み、CPUs と記憶ディスクとの間のデータの転送中に類似するデータ・コピーを除去することができるシステムを提供する。

[0010]

[課題を解決するための手段]

本発明の上記目的は、データを転送するデータ処理システムであって、要求 CPU を含んでいる複数の中央処理装置(CPU);CPU の一つが記憶装置へのアクセスを制御する少なくとも一つの記憶装置;複数の CPUs と記憶装置とを相互接続するネットワーク;要求 CPU による記憶装置の直接アクセスを供給し、要求 CPU のバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に仮想メモリ・アドレスを、記憶装置へのアクセスを制御している CPUs の一つに供給する要求 CPU における手段、ワーク要求に応じておりかつネットワークを通して CPUs の一つと直接インターフェイスしている記憶装置に仮想メモリ・アドレスを含んでいるワーク要求を送る CPUs の一つにおける手段、を含んでいるアクセス手段;を備え、データは、要求 CPU と記憶装置との間で直接転送されるデータ処理システムによって達成される。

[0011]

本発明のデータ処理システムでは、直接アクセスは、要求 CPU に記憶装置から仮想メモリ・アドレスにおけるバッファ・メモリの中にデータを読取らせるようにしてもよい。

本発明のデータ処理システムでは、データは、それぞれが、宛先ノード識別、ソース・ノード識別、仮想メモリ・アドレス、及び複数のデータ・ワードを含んでいるデータ・パケットで送信されるようにしてもよい。

With step 2, as for disk treatment 130, data which is arranged in buffer 150 in disk 100 entry \*.

Because with step 3, transfer of data to disk 100 ended, as for disk treatment 130, you respond to request treatment 145.

[0009]

With this configuration, as for data, copy it makes intermediate CPU which possesses related disk treatment from CPU which possesses request treatment and from intermediate CPU copy makes disk.

throughout, of transfer of data between CPUs and storage disc it is desirable to remove data \*copy which resembles.

You consider objective of this invention, to problem of the above-mentioned Prior Art, you offer system which can remove CPUs and data \*copy which resembles while transferring data between the storage disc.

[0010]

[Means to Solve the Problems]

As for above-mentioned objective of this invention, with data processing system which transfers data, central processing unit of plural which includes therequired CPU (CPUs); storage device to supply access directly with CPUs of storage device; plural of at least one where one of CPUs controls access to storage device and network; required CPU which interconnect does storage device, the hypothetical memory \*address for buffer of required CPU only formation with storage device access request hypothetical memory \*address, We to have responded to means. work request in required CPU which is supplied to one of CPUs which controls access to the storage device to have access means; which includes means. in one of the CPUs which sends work request which includes hypothetical memory \*address in storage device which interface has been made one of the CPUs directly and through network, as for data, it is achieved with required CPU and data processing system which is directly transferred between storage device.

[0011]

With data processing system of this invention, directly as for access, data is made to grasp in buffer \*memory in required CPU from storage device in the hypothetical memory \*address it is possible to require.

With data processing system of this invention, as for data, each one, is transmitted with data \*packet which includes addressee node identification, source \*node identification and hypothetical memory \*address, and data \*word of plural may require.

本発明のデータ処理システムでは、記憶装置は、転送のためのデータを含んでいるデータ・パケットのそれぞれが要求 CPU に送信された後で CPUs の一つにアドバイスを手段を含むようにしてもよい。

本発明のデータ処理システムでは、直接アクセスは、要求 CPU に、データの転送のために仮想メモリ・アドレスにおけるバッファをアクセスしている記憶装置の中に書込ませるようにしてもよい。

[0012]

本発明のデータ処理システムでは、アクセス手段は、要求 CPU のバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に仮想メモリ・アドレスを CPUs の一つに供給する少なくとも一つのソフトウェア・ルーチンを含むようにしてもよい。

[0013]

[作用]

本発明は、データを転送するためのデータ処理システムを供給する。

このシステムは、ネットワークによって相互接続された中央処理装置(CPU)及び記憶装置を含む。

CPUs は、要求処理及び記憶処理(ディスク処理とも呼ばれる)を含む。

記憶処理は、記憶装置へのアクセスを制御する。

ソフトウェア・ルーチンは、要求 CPU(要求処理を含んでいる CPU)による記憶装置への直接アクセスを供給するために用いられる。

要求 CPU のバッファに対するサーバ・ネット仮想メモリ・アドレスは、要求 CPU に生成される。

記憶装置アクセス要求と共にサーバ・ネット仮想メモリ・アドレスは、記憶処理を含んでいる CPU に送られる。

サーバ・ネット仮想メモリ・アドレスを含んでいるワーク要求は、記憶処理から記憶装置に送られる。

次いでデータは、要求 CPU と記憶装置との間に直接転送される。

記憶装置は、次いで、ワーク要求に応答する。

[0014]

本発明の更なる態様及び特徴は、添付した図

With data processing system of this invention , as for storage device , after each one of the data \*packet which includes data for transferring was transmitted to therequired CPU means which does advise in one of the CPUs is included, it is possible to require.

With data processing system of this invention , directly as for access , in therequired CPU , entry it can increase in storage device which access does buffer in hypothetical memory \*address for transferring data itis possible to require.

[0012]

With data processing system of this invention , as for access means , hypothetical memory \*address for buffer of required CPU only formation with storage device access request software \*routine of at least one which supplies hypothetical memory \*address to the one of CPUs is included, it is possible to require.

[0013]

[Working Principle ]

this invention supplies data processing system in order to transfer data .

this system includes central processing unit which interconnect is done (CPUs ) and storage device with network .

CPUs includes request treatment and recording routine (Also disk treatment is called ) .

recording routine controls access to storage device .

software \*routine is used in order to supply directly access to storage device with required CPU (CPU which includes request treatment) .

server \*network hypothetical memory \*address for buffer of required CPU isformed to required CPU .

With storage device access request server \*network hypothetical memory \*address is sent to CPU which includes recording routine .

work request which includes server \*network hypothetical memory \*address is sentto storage device from recording routine .

Next data is transferred to required CPU and between the storage device directly .

As for storage device , next, you respond to work request.

[0014]

In person skilled in the art it becomes clear by fact that further

面に関して取入れられるべき、本発明の以下の詳細の説明を読むことにより当業者に明らかになるであろう。

【0015】

【実施例】

本発明は、プロセッサ及び I/O 通信に対する接続性を供給するネットワークを介して高信頼システムにおける直接バルク・データ転送を供給する。

この直接データ転送は、要求処理を走らせている CPU とディスク処理を走らせている CPU との間のデータの複写を除去すると同時にディスク処理ペア構成を維持する。

それゆえに、データは、要求処理のバッファと記憶装置との間で直接複写される。

結果として、(1)データがディスク処理のバッファに転送されないでネットワーク帯域幅は、セーブされ、(2)ディスク処理がそのバッファの中にデータを受け取らなくともよいのでコンテキスト・スイッチ時間がセーブされ、(3)バッファ複写が回避されるので、入力/出力(I/O)ラテンシーは、低減され、かつ(4)メッセージ・システムが要求処理とディスク処理との間でデータを送らないのでワークは、メッセージ・システムから“負担が軽減される”。

このメッセージ・システムは、プロセッサ間通信に用いられる。

【0016】

図 3 は、コントローラが“プッシング”(pushing) しているときに発生するデータの転送を示している。

例えば、データがディスク 105 から CPU メモリへ転送されるときに、サーバ・ネット・アダプタ 30 は、CPU メモリにデータ 170 を“プッシング”する。

これは、また、“読取り”オペレーションとも呼ばれる。

この構成では、要求処理及びデータ処理の両方は、CPU 326 に配置されている。

この CPU 326 は、サーバ・ネット・アダプタ 30 に、データ転送に関連付けられたパラメータ(例えば、遠隔ノード識別)と共に、データ 170 を受取る、バッファ 162 に対するサーバ・ネット仮想アドレス 172 を送る。

サーバ・ネット仮想アドレス 172 は、バッファ 162 の物理アドレスを取得するために用いる情報を

embodiment and feature of this invention should adopt in regard to drawing which is attached, explanation of details below this invention read.

[0015]

[Working Example (s)]

As for this invention, through network which supplies connectivity for the processor and I/O communication bulk \*data transfer in high reliability system is supplied directly.

this directly data transfer when copy of data between the CPU which is making CPU and disk treatment run which are making request treatment run is removed maintains disk treatment pair configuration simultaneously.

Consequently, data copy is done directly buffer of request treatment and between storage device .

As result, because (1) data is not transferred to buffer of disk treatment, because network bandwidth to be done save , (2) disk treatment in buffer data also not receiving, it is good, because context \*Switch time save to be done, (3) buffer copy is evaded, the input/output (I/O) [latency] is decreased, At same time because (4) message \*system does not send data during request treatment and disk treatment, as for work , from the message \*system "burden is lightened".

this message \*system is used for interprocessor communication .

[0016]

Figure 3 , when controller " [pushing] " being (pushing) , has shown transfer of data which occurs.

When for example data is transferred from disk 105 to CPU memory , server \*network \*adapter 30 "[pushing] " does data 170 in CPU memory .

As for this, in addition, " also reading "operation is called.

With this configuration , as for both of request treatment and data processing , it is arranged in CPU 326.

As for this CPU 326, in server \*network \*adapter 30, with parameter (for example remote node identification) which relation is attached to data transfer , data 170 is received, server \*network virtual address 172 for buffer 162 is sent.

server \*network virtual address 172 includes data which is used in order to acquire the physical address of buffer 162.

含む。

このサーバ・ネット仮想アドレス 172 は、後ほど物理アドレスの中に変換して戻される。

好ましい実施例では、サーバ・ネット仮想アドレス 172 は、頁番号と、関連物理アドレスを決定するために一緒に用いられることが必要であるオフセットとの両方を含む。

従って、二つ以上のサーバ・ネット仮想アドレス 172 を一つのバッファに対して設けることができる。

サーバ・ネット仮想アドレスは、全ての目的に対して参考文献として採り入れられた添付特許願により詳細に開示されている。

バッファ 162 は、要求処理に属する。

転送に関連付けられたパラメータは、“ワーク要求”(work request)を介して CPU3 26 からサーバ・ネット・アダプタ 30 に送られる。

この情報で、サーバ・ネット・アダプタ 30 は、I/O デバイス(例えば、\$F ディスク 105)から要求処理のメモリに直接データ 170 を転送することができる。

同様な構成で、CPU は、I/O デバイス(ディスク)に“書込む”ことができる。

これは、サーバ・ネット・アダプタが CPU メモリからデータを“プルング”(pulling)している呼ばれる。

この構成では、CPU3 26 は、I/O 空間に関連付けられたパラメータと共に、データを含んでいる CPU バッファ 162 に対するサーバ・ネット仮想アドレス 172 をサーバ・ネット・アダプタ 30 に送る。

再び、これらのパラメータは、“ワーク要求”に含まれる。

この情報で、サーバ・ネット・アダプタ 30 は、CPU メモリからディスク・デバイス 105 にデータを転送することができる。

従って、データは、非ディスク処理を有する CPU と、関連ディスク処理を走らせている CPU を通じてデータを転送しないディスクとの間で転送される。

[0017]

サーバ・ネット雲を通して送られたパケット(サーバ・ネット・パケットと呼ばれる)は、サーバ・ネット・パケットに対するソース・ノード 174 及び宛先ノード 176 を識別するヘッダを含む。

used in order to acquire the physical address of buffer 162.

this server \*network virtual address 172 is reset, about after converting in physical address .

With desirable Working Example , as for server \*network virtual address 172, page number and offset whose it is necessary to be used together in order to decide therelated physical address both is included.

Therefore, it is possible to provide server \*network virtual address 172 of two or more vis-a-vis buffer of one .

As for server \*network virtual address , vis-a-vis all objective it is disclosed in detail by theattachment patent application which is adopted Cited Reference (s) as.

buffer 162 belongs to request treatment.

parameter which relation is attached to transfer, through "work required " (workrequest ), is sent to server \*network \*adapter 30 from central processing unit 326.

With this data , server \*network \*adapter 30 directly from I/O device (for example \$Fdisk 105 ) can transfer the data 170 to memory of request treatment.

With similar configuration , " entry \* " thing can designate CPU , as I/O device (disk ) .

As for this, server \*network \*adapter from CPU memory data "[puringu ] " it is (pulling ), it is called.

With this configuration , as for CPU 326, with parameter which relation is attached to I/O space , server \*network virtual address 172 for CPU buffer 162 which includes the data is sent to server \*network \*adapter 30.

Again, these parameter are included "work request " .

With this data , server \*network \*adapter 30 from CPU memory can transfer data to the disk \*device 105.

Therefore, as for data , it is transferred between disk whichdoes not transfer data through CPU which is making the CPU and related disk treatment run which possess non- disk treatment.

[0017]

packet (server \*network \*packet it is called ) which is sent through server \*network cloud includes header which identifies source \*node 174 and addressee node 176 for server \*network \*packet .



この識別は、ノード識別(ID)の形である。

好ましい実施例では、CPU<sub>s</sub> 及び他の周辺デバイスだけがノード ID を有する。

各サーバ・ネットワーク・パケット・ヘッダは、また、パケットの型を示すトランザクション型フィールドを含む。

パケット型が読取られるかまたは書込まれるときに、パケットは、宛先ノードから肯定応答(acknowledgement) を導き出す。

読取りパケットに対して、この肯定応答は、読取りを実行しているノードにリターンされるべきデータを含む。

書込みパケットに対して、この肯定応答は、書込みオペレーションに対する成功または失敗状態をリターンする。

[0018]

サーバ・ネットワーク読取り/書込みパケットで指定されたサーバ・ネットワーク・アドレスは、宛先ノードにおける物理アドレスではなく、その代わり、許可チェックされそして、許可チェックがパスしたならば、物理アドレスに変換される、アドレスである。

このサーバ・ネットワーク・アドレスは、上記で参照したサーバ・ネットワーク仮想アドレスである。

許可検査は、例えば、ソース・ノード識別妥当性検査、変換型検査(即ち、読取り/書込み許可)及びバウンド(bounds)検査の形である。

アドレス妥当性検査及び変換(AVT)表は、サーバ・ネットワーク仮想アドレスに適用された変換及び許可検査に用いられる。

ソース・ノード ID を確認(検査)するために、アクセスした AVT エントリのソース ID フィールドは、用いられる AVT エントリに対応するソースを指定する。

このソース ID フィールドは、不整合(mismatch) がアクセスを否定する AVT エラー割込みを結果として生ずるように、要求しているメッセージ・パケットに含まれるソース ID と比較される。

トランザクション型検査は、読取りまたは書込みのためのアクセスが許可されるかどうかを決定するための検査を含む。

バウンド検査に対して、AVT エントリの下部バウンド・フィールド及び上部バウンド・フィールドは、アクセスが許可されるかどうかを決定するためにオフセット値と比較される。

this identification is shape of node identification (ID).

With desirable Working Example , just CPUs and other peripheral device have the node ID.

As for each server \*network \*packet \*header , in addition, transaction type field which shows the type of packet is included.

packet type is grasped or or entry rare \* time, packet deduces affirmative response (acknowledgement ) from addressee node .

Vis-a-vis reading packet , this affirmative response includes data which return it should you make node which executes reading.

Vis-a-vis entry packet , this affirmative response return does successor failure state for entry operation .

[0018]

server \*network reading / as for server \*network \*address which is appointed with entry packet , it was not a physical address in addressee node , substituting and grant check it was done and and, grant check did pass , if is, it is converted to physical address , it is a address .

this server \*network \*address is server \*network virtual address which description above refers to.

grant inspection, for example source \*node identification adequacy inspection and conversion type checking (Namely, reading / entry grant ) and is shape of bound (bounds ) inspection.

address adequacy inspection and conversion (AVT ) chart is used for conversion and grant inspection which are applied to server \*network virtual address .

source which corresponds to AVT entry which as for source ID field of the AVT entry which access is made in order you verify (Inspection) source \*node ID, is used is appointed.

this source ID field is compared as occurred with AVT error interruption where mismatch (mismatch ) negative does access as result, source ID which is included in message \*packet which is been required.

transaction type checking, includes inspection in order to decide whether or not access for reading or entry can be done grant .

Vis-a-vis bound inspection, bottom bound \*field and upper part bound \*field of AVT entry , the offset value are compared in order to decide whether or not access is done grant .

この型の許可検査は、添付特許出願(例えば、その頁 55-61、図 13A-13C)に詳細に記載されている。

[0019]

データの転送を達成するために用いられるハードウェア直接メモリ・アクセス(DMA)エンジンは、また、ブロック転送エンジンとも呼ばれる。

好ましい実施例では、ブロック転送エンジンは、非同期でメモリ・バッファ上でチェックサムを計算することができ、かつ少なくとも一つの DMA エンジンが各 CPU に配置される。

上記したように、チェックサムは、転送されるデータのブロックに対してデータの一貫性を確実にする。

上記したように、CPU20、22、24 または 26 は、サーバ・ネットワークにわたりデータのペケットを転送するためにブロック転送エンジンにバッファを委ねることができる。

更に、CPU20、22、24 または 26 は、チェックサムを計算しかつ別のバッファにチェックサムを配置するためにブロック転送エンジンにバッファを委ねることができる。

そして、好ましい実施例では、SCSI テープ 40 に配置された DMA エンジンは、(ディスク"読取り"または"書込み"に対して)各データ転送に対して"プッシング"(pushing)または"プルング"(pulling)を実行する。

[0020]

上記したように、本発明の直接データ転送が行われるときに、データは、要求発生 CPU とディスク処理を含んでいる CPU との間で転送されない。

代わりに、サーバ・ネットワーク仮想アドレスが生成されかつ用いられる。

このサーバ・ネットワーク仮想アドレスは、直接データ転送を実行するためにサーバ・ネットワーク・アダプタによって用いられる。

再び、この処理において二つの型のデータ転送が存在する。

第 1 は、サーバ・ネットワーク仮想アドレスを生成すること及び用いることを含むディスク書込みオペレーションである。

サーバ・ネットワーク・アダプタは、ディスク書込みに対して必要なオペレーションを行うためにこのサーバ・ネットワーク仮想アドレスを用いる。

grant inspection of this type is stated in detail in attachment patent application (page 55-61, Figure 13 A-&lt;SP&gt;13&lt;/SP&gt;C of for example ).

[0019]

hardware which is used in order to achieve transfer of data directly memory \*access (DMA ) engine in addition, also block transfer engine is called.

With desirable Working Example , as for block transfer engine , it is possible with the asynchronous to calculate checksum on memory \*buffer , at same time DMA engine of at least one is arranged in each CPU .

As inscribed, checksum makes consistency of data secure vis-a-vis block of data which is transferred.

As inscribed, CPU 20, 22, 24 or to entrust buffer to block transfer engine in orderto transfer packet of data over server \*network cloud it is possible26.

Furthermore, CPU 20, 22, 24 or as for 26, only calculation to entrust the buffer to block transfer engine in order classified by to arrange checksum in the buffer it is possible checksum .

And, with desirable Working Example , as for DMA engine which is arranged in SCSI tip 40, " [pusshingu ] " (pushing ) or " [puringu ] " (pulling ) is executed (disk \* reading " or " entry " confronting ) vis-a-vis each data transfer .

[0020]

As inscribed, when this invention data transfer is done directly, data isnot transferred between CPU which includes requiredoccurrence CPU and disk treatment.

server \*network virtual address is formed by substituting and, and is used.

this server \*network virtual address in order directly to execute data transfer is used with the server \*network \*adapter .

Again, data transfer of type of two exists at time of this treating.

first is disc entry operation which includes fact that the server \*network virtual address is formed and fact that it uses.

server \*network \*adapter uses this server \*network virtual address in order to do necessary operation vis-a-vis disk entry.

それゆえに、(要求処理バッファに対する)サーバ・ネット仮想アドレスに対する AVT エントリは、要求処理バッファをアクセスすることを必要とするサーバ・ネット・アダプタに対するサーバ・ネット・ノード ID を指定する。

[0021]

ディスク書き込みオペレーションに対して、要求処理を走らせている発生 CPU は、ディスク処理への要求を発行する前にデータのチェックサムをまず計算する。

要求処理がそのバッファに対してサーバ・ネット仮想アドレスを生成した後、要求処理は、そのサーバ・ネット仮想アドレスをディスク処理にパスする。

ディスク処理は、ソース ID 妥当性検査が失敗しうるので、要求処理バッファをアクセスするためにこのサーバ・ネット仮想アドレスを用いることができない。

サーバ・ネット仮想アドレスを受け取った後、ディスク処理は、(1)要求を発生している CPU(要求処理を含んでいる CPU)に対するサーバ・ネット・ノード ID への処理、及び(2)要求処理に関連付けられたバッファに対するサーバ・ネット仮想アドレス、を指定しているサーバ・ネット・アダプタへ“ワーク要求”を送る。

サーバ・ネット・アダプタは、それが書き込みオペレーションであるので、サーバ・ネット仮想アドレスを用いてデータを次いで“プル”(pulls)する。

このデータの転送の終りで、(一般に割込みの形の)通知が、ディスク処理を含んでいる CPU に送信して戻される。

次いで、ディスク処理は、データ転送の終了をそれに通知するために要求処理に応答する。

AVT 表及びサーバ・ネット仮想アドレスに関する更なる情報は、添付特許出願に供給されている。

[0022]

第 2 の型のデータ転送は、ディスク読取りに対してである。

図 4 は、ディスク読取りに対する直接データ転送の一例を示している。

段階 11 では、バッファ 150 に対するサーバ・ネット仮想アドレスが生成される。

バッファ 150 は、要求処理 146 に関連付けられる。

Consequently, AVTentry for (In required treatment buffer it confronts ) server \*network virtual address appoints server \*network \*node ID for the server \*network \*adapter which needs that required treatment buffer is done access .

[0021]

Vis-a-vis disk entry operation , occurrence CPU which is makingrequest treatment run before issuing demand for disk treatment,calculates checksum of data first.

Request treatment after forming server \*network virtual address vis-a-vis buffer , requesttreatment pass makes server \*network virtual address disk treatment.

Because as for disk treatment, source IDadequacy inspection can fail, this server \*network virtual address cannot be used in order access to do required treatment buffer .

After receiving server \*network virtual address , as for disk treatment, "work request "you send to server \*network \*adapter which appoints server \*network virtual address , for buffer whichrelation is attached to treatment, and (2) request treatment to the server \*network \*node ID for CPU (CPU which includes request treatment) which generates (1) request.

Because as for server \*network \*adapter , that is entry operation , data " [puru ] " (pulls )next making use of server \*network virtual address .

With end of transfer of this data , (Generally shape of interruption ) notification, transmitting to CPU which includes disk treatment, it is reset.

Next, as for disk treatment, you respond to request treatment inorder to notify end of data transfer to that.

AVT chart and further data regarding server \*network virtual address are suppliedto attachment patent application .

[0022]

data transfer of second type is vis-a-vis disk reading.

Figure 4 has shown one example of data transfer for disk reading directly.

With step 11, server \*network virtual address for buffer 150 is formed.

buffer 150 relation is attached to request treatment 146.

段階 12 では、要求処理 146 は、要求をディスク (Dp2) 処理 133 へ送る。

段階 13 では、ディスク処理 133 は、ワーク要求をサーバ・ネット・アダプタ 30 へワーク要求を送る。

段階 14 では、サーバ・ネット・アダプタ 30 は、SF ディスク 105 からバッファ 150 へのデータの転送を実行する。

段階 14 は、要求データの全てがバッファ 150 へ送られるまで行われる。

データ転送が終了したときには、サーバ・ネット・アダプタ 30 は、段階 15 でディスク処理 133 に割り込む。

段階 16 では、ディスク処理 133 は、データ転送が終了したことを要求処理 146 に知らせる。

段階 17 では、要求処理を走らせている発生 CPU は、バッファのチェックサムを計算しそしてデータを受け入れる前にそれを確認する。

[0023]

ディスク書込みに対する直接データ転送は、(1) 要求がディスク処理に送られる前にチェックサムが要求処理によって計算され、かつ(2)データの転送の方向が CPU メモリからディスクへであることを除き、ディスク読取りに非常に類似している。

図 5 は、ディスク書込みに対する直接データ転送の一例を示している。

段階 21 では、要求処理 146 は、チェックサム計算を実行する。

段階 22 では、要求プロセッサ 146 は、バッファ 150 に対しかつチェックサム・バッファに対してサーバ・ネット仮想アドレスを生成する。

段階 23 では、要求処理 146 は、その要求をディスク処理 133 へ送る。

段階 24 では、ディスク処理 133 は、ワーク要求をサーバ・ネット・アダプタ 30 へ送る。

段階 25 では、サーバ・ネット・アダプタ 30 は、バッファ 150 からのデータ及びチェックサム・バッファからのチェックサムを SF ディスク 105 へ転送する。

要求したデータの全てがバッファ 150 からディスク 105 へ転送された後、サーバ・ネット・アダプタ 30 は、段階 26 でディスク処理 133 に割り込む。

段階 27 では、ディスク処理 133 は、データ転送

With step 12, as for request treatment 146, request is sent to the disk (Dp2) Process 133 .

With step 13, as for disk Process 133 , work request work request is sent to server \*network \*adapter 30.

With step 14, as for server \*network \*adapter 30, transfer of data to buffer 150 is executed from SFdisk 105.

step 14 is done, until all of required data is sent to buffer 150.

When data transfer ends, it interrupts server \*network \*adapter 30, disk Process 133 with the step 15.

With step 16, as for disk Process 133 , it informs about fact that the data transfer ends request treatment 146.

With step 17, as for occurrence CPU which is making request treatment run, it calculates checksum of buffer and and before accepting data , it verifies that.

[0023]

Directly as for data transfer for disk entry, before (1) request is sent to disk treatment, checksum is calculated in request treatment, resembles to unusual in disk reading at same time direction of transfer of (2) data from CPU memory excluding fact that is to disk .

Figure 5 has shown one example of data transfer for disk entry directly.

With step 21, as for request treatment 146, checksum calculation is executed.

With step 22, as for required processor 146, server \*network virtual address is formed vis-a-vis confronting and checksum \*buffer in buffer 150.

With step 23, as for request treatment 146, that request is sent to disk Process 133 .

With step 24, as for disk Process 133 , work request is sent to server \*network \*adapter 30.

With step 25, as for server \*network \*adapter 30, data from buffer 150 and the checksum from checksum \*buffer are transferred to SFdisk 105.

After all of data which it requires was transferred from buffer 150 to disk 105, it interrupts server \*network \*adapter 30, disk Process 133 with the step 26.

With step 27, as for disk Process 133 , it informs about fact

が終了したことを要求処理 146 に知らせる。

[0024]

直接データ転送に対するソフトウェアは、ディスク・デバイスからの(または、に)要求処理を走らせている CPU に(または、から)データを直接送らせる。

好ましい実施例において直接データ転送を実行するために、デバイス・ハンドルが用いられる。

このデバイス・ハンドルは、遠隔サーバ・ネット・ノード ID をデータ転送に対する AVT エントリに供給する。

このノード ID によって識別されたサーバ・ネット・アダプタのみがデータ転送に対する AVT 表におけるエントリをアクセスすることができる。

AVT 表は、それが正しいノード ID を有するサーバ・ネット・アダプタにアクセス可能であるように、要求 CPU のバッファをサーバ・ネット仮想アドレス空間にマップする。

起動されたときに、デバイス・ハンドルは、ノード ID を含むデータ転送パラメータを指し示す。

このデバイス・ハンドルを取得するために、“TSER\_DEVINSTALL”とレッテルを貼られたサーバ・ネット・ノード・ルーチンが好ましい実施例で用いられる。

従って、遠隔サーバ・ネット・ノードに関連付けられたパラメータは、このルーチンが呼出されるときにコードに指定される。

[0025]

好ましい実施例では、システムは、データ転送に対する要求が認識されるときに直接データ転送を起動しかつ TSER\_DEVINSTALL を実行する。

直接データ転送処理が終了したときに、デバイス・ハンドルは、例えば、TSER\_DEVREMOVE ルーチンを呼出すことによってリターンされる。

第 2 の実施例では、デバイス・ハンドル変換キャッシュに対する汎用(グローバル)デバイス・ハンドルまたはスタック/モジュール/スロットが各 CPU に配置される。

この実施例では、TSER\_DEVINSTALL 及び TSER\_DEVREMOVE ルーチンは、呼び出されない。

ユーザは、多くの方法で直接データ転送を用いることの要望を示すことができる。

that the data transfer ends request treatment 146.

[0024]

Directly, software for data transfer can delay (Or, empty) data directly in CPU which is making (Or) request treatment from disk \*device run.

In order directly to execute data transfer in desirable Working Example, it can use device \*handle.

remote server \*network \*node ID it supplies this device \*handle, to AVTentry for data transfer.

entry in AVT chart only server \*network \*adapter which is identified with this node ID for data transfer access is possible.

AVT chart as been a accessible in server \*network \*adapter which possesses node ID where that is correct, map designates required CPU's buffer as the server \*network virtual address space.

When being started, device \*handle indicates data transfer parameter which includes the node ID.

In order to acquire this device \*handle, it is used with Working Example where the server \*network \*node \*routine which was pasted "TSER\_DEVINSTALL" and label is desirable.

Therefore, parameter which relation is attached to remote server \*network \*node, when the this routine is called, is appointed to cord.

[0025]

With desirable Working Example, as for system, when request for the data transfer is recognized, data transfer only starting TSER\_DEVINSTALL is executed directly.

Directly when data transferring process ends, device \*handle return is done by fact that for example TSER\_DEVREMOVE routine is called.

With second Working Example, general purpose (global) device \*handle or stack /module /slot for device \*handle conversion cache is arranged in each CPU.

With this Working Example, as for TSER\_DEVINSTALL or TSER\_DEVREMOVE routine, it is not called.

To show demand of thing which directly uses data transfer with themany method it is possible user.

例えば、SETMODE 141 は、ディスク・ファイル転送に対する大きなデータ転送モードをイネールできる。

この SETMODE は、ユーザに、ディスク処理キャッシュをバイパスするディスク・ファイルへの大きな、非構成アクセスを彼らが要望することを指定されることができる。

別の例では、BULKREAD 及び BULKWRITE は、バックアップ、復元、ダンプ、等が要求されるときに、内部ツールとして用いることができる。

BULKREAD 及び BULKWRITE は、また、ディスク処理キャッシュもバイパスする。

それゆえに、BULKREAD または BULKWRITE 或いは SETMODE 141 がファイル転送に対してイネールされたならば、直接データ転送は、転送タスクを実行するために用いられる。

[0026]

好ましい実施例では、SETMODE 141 が示されたときに、メッセージは、そのキャッシュをフラッシュするためにディスク処理に送られる。

この SETMODE 要求への応答の一部として、ディスク処理は、それが直接データ転送を支持することができるファイル・システムに対して指示(indication)をリターンする。

ディスク処理は、また、二つのサーバ・ネット IDs をファイル・システムにリターンする。

これらのサーバ・ネット IDs は、関連ディスク及びそのミラー・ディスクへの二つの主要(1 次)経路を構成する。

ディスク・デバイスへの書き込みを行うときに両方のサーバ・ネット ID が用いられ、かつ一つのサーバ・ネット ID のみが読取りオペレーションに対して用いられる。

ファイル・システムがディスク処理から応答を受け取るときに、それは、応答を解読しかつサーバ・ネット・コードを二つのサーバ・ネット IDs に設置するために直接データ転送ルーチンを呼出す。

[0027]

同様に、BULKREAD または BULKWRITE が指示されたときに、ファイル・システムは、通常のパルク・データ転送要求をディスク処理に送る。

可能な場合には、ディスク処理は、それが直接データ転送を支持することができることを示して

Large data transfer mode where for example SETMODE141 confronts disk \*file transfer enable is possible.

this SETMODE is large to disk \*file which bypass does disk treatment cache to user , non- configuration access they to demand it impossible to be appointed.

With another example, when backup , restoration and dump , etc are required, you can use BULKREAD and BULKWRITE, as interior tool .

As for BULKREAD and BULKWRITE, in addition, bypass it does also disk treatment cache .

Consequently, BULKREAD or BULKWRITE or SETMODE141 enable it was done vis-a-vis file transfer if is, directly data transfer is used in order to execute transfer task .

[0026]

With desirable Working Example , when SETMODE141 is shown, as for message ,it is sent to disk treatment in order flash to do cache .

As portion of response to this SETMODE request, disk treatment return does display (indication ) vis-a-vis file \*system where that directly can support data transfer .

disk treatment, in addition, server \*network IDs of two return it designates as file \*system .

These server \*network IDs configuration do principal (primary ) path of two to the related disk and its mirror \*disk .

When doing entry seeing to disk \*device , it can use server \*network ID of the both , at same time it can use only server \*network ID of one vis-a-vis reading operation .

When file \*system receives response from disc treatment, that response only reading calls data transfer routine in order to install server \*network \*cord in the server \*network IDs of two directly.

[0027]

When in same way, BULKREAD or BULKWRITE display being done, the file \*system sends conventional bulk \*data transfer request to disk treatment.

In possible case, as for disk treatment, you respond to file \*system which has shown fact that that directly can support

いるファイル・システムに応答する。

この時には、ディスク処理は、また、関連ディスク及びそのミラーへの二つの主要経路を構成する二つのサーバ・ネット ID をリターンする。

次いで、ファイル・システムは、後続のバルク・データ転送呼出しに対して直接データ転送を用い、かつまたサーバ・ネット・コードを二つのサーバ・ネット ID に設置するために直接データ転送ルーチンを呼出す。

(二つサーバ・ネット ID を設置する)直接データ転送セッション確立の一部として、直接データ転送ルーチンは、また、FLEXPOOL からの転送情報ブロック(TIB)に対する空間も獲得する。

TIB は、直接データ転送ルーチンとサーバ・ネット・コードとの間のインターフェイスに用いられる。

TIB は、また、チェックサム計算を行うためにサーバ・ネット・コードが DMA エンジンを用いるということを要求するために直接データ転送ルーチンによっても用いられる。

好ましい実施例では、チェックサム・オペレーションが同期であるので、単一の TIB が用いられる。

FLEXPOOL は、メモリ管理に対して空間を割り当てる。

【0028】

図 6 は、どのように二つのサーバ・ネット ID が、ディスク及びそのミラー・ディスクへの二つの経路に対して用いられるかを示している。

好ましい実施例では、二つのサーバ・ネット ID は、SCSI チップ 40 及び 42 に関連付けられる。

図 6 に示すように、これらの SCSI コントローラチップ 40 及び 42 は、ディスク 100 及びそのミラー・ディスク 115 へのアクセスを供給する。

好ましい実施例では、フォールト・フリー(無欠陥)システムは、次のものを備えている:CPU0 20 及び CPU1 22;二つのサーバ・ネット 10 及び 12;ディスク及びそのミラー・ディスク 14 への 1 次(主要)経路及びディスク及びそのミラー・ディスク 16 へのバックアップ経路;サーバ・ネット・アダプタ 30 及び 32;SCSI チップ 40、42、44 及び 46;ディスク 100 及びそのミラー・ディスク 115;チェーン 120 及び 122。

この構成では、システムの全てのコンポーネントは、バックアップとして少なくとも一つの冗長コンポーネントを有する(例えば、ディスク 100 は、バックアップ・ミラー・ディスク 115 を有する)。

data transfer .

At time of this , as for disk treatment, in addition, the related disk and server \* [neeto ] ID of two which configuration does principal path of two to its mirror are done return .

Next, calls file \*system at same time and data transfer routine in order to install server \*network \*cord in server \*network ID of two directly vis-a-vis the succeeding bulk \*data transfer call making use of data transfer directly.

(two server \*network ID is installed ) As portion of data transfer session establishment directly, directly data transfer routine in addition, it acquires also space for transmitted information block (TIB ) from FLEXPOOL.

TIB directly is used for interface between data transfer routine and the server \*network \*cord .

As for TIB, in addition, in order to require that server \*network \*cord uses DMA engine in order to calculate checksum it is used directly even with data transfer routine .

Because with desirable Working Example , checksum \*operation is synchronization , it can use single TIB.

FLEXPOOL allots space vis-a-vis memory managing .

【0028】

Which way, it has shown Figure 6 , whether it can use server \*network ID of the two , vis-a-vis path of two to disk and its mirror \*disk .

With desirable Working Example , as for server \*network IDs of two , relation it is attached to SCSI tip 40 and 42.

As shown in Figure 6 , these SCSI controller \*tip 40 and 42 supplies access to the disk 100 and its mirror \*disk 115.

With desirable Working Example , [fooruto ] \* as for free (defect-free ) system , server \*network cloud 10 of the:CPU 020 and CPU 122;two which have following ones and 12; the backup path ;server \*network \*adapter 30 to primary (Principal) path and disk and its mirror \*disk 16 to disk and its mirror \*disk 14 and 32; SCSI tip 40, 42, 44 and 46; disk 100 and its mirror \*disk 115;chain 120 and 122.

With this configuration , as for all component of system , it possesses redundant component of at least one as backup , (for example disk 100 has backup \*mirror \*disk 115 ) .

ックアップ・ミラー・ディスク 115 を有する)。

従って、データ書き込みオペレーションの間中、全てのデータ転送は、エラーが発生しかつ正しいデータが一つのディスクからアクセスすることができないならば、そのデータもまた、アクセスのために別のディスクに配置されるように二つのディスク(ディスク及びそのミラー・ディスク)に対してなされる。

この構成は、もしあれば、データの汚染を結果としてほとんど生じない。

ディスク読み取りオペレーションの間中、直接データ転送ソフトウェアは、ディスク 100 または 115 の一つのみによってアクセスのためにデータ・バッファをマップする。

それゆえに、バッファは、一つのサーバ・ネットワーク仮想アドレス空間の中にだけマップされる。

[0029]

代替実施例では、4 つのサーバ・ネットワーク ID が直接データ転送のために用いられる。

この構成では、データ・バッファは、全ての 4 つの SCSI コントローラ・チップ 40、42、44 及び 46 によるアクセスに対するサーバ・ネットワーク仮想アドレス空間の中に冗長的にマップされなければならない。

これは、ディスク処理に SCSI コントローラ・チップ 40、42、44 または 46 に対してデータ転送要求を発行させる。

直接データ転送支援ソフトウェアは、ソフトウェア・ルーチンのライブラリとして構成される(ルーチンと呼ばれる)。

好ましい実施例では、ファイル・システムは、これらルーチンへのクライアントである。

それゆえに、ファイル・システムが直接データ転送が起動されることを決定したときに、それは、バッファをマップしかつチェックサム計算を行うために必要な動作を起こすルーチンを呼出す。

直接データ転送ルーチンが行われたときに、ファイル・システムは、メッセージをディスク処理を含んでいる適切な CPU に送るためにメッセージ・システムを用いる。

[0030]

図 7 は、直接データ転送を有するシステム・ソフトウェア階層化の一例である。

図 7 に示すように、ファイル・システム 210 は、直接メッセージ・システム 230 を呼出すことができ

Therefore, as for throughput, all data transfer of data writing seeing operation, error only occurrence correct data can do access from disk of the one, if is, in order data and because of access to bearranged in another disk, you can do vis-a-vis disk (disk and its mirror \*disk) of the two.

this configuration, if it is, does not occur for most part with the pollution of data as result.

throughout, of disk reading operation data transfer software disk 100 or with only one of 115 map does data \*buffer directly because of the access.

Consequently, buffer map is done to just in server \*network virtual address space of the one.

[0029]

With alternate embodiment, it can use server \*network ID of 4 directly because of the data transfer.

With this configuration, as for data \*buffer, SCSI controller \*tip 40, 42, 44 of all 4 and map you must make redundant in server \*network virtual address space for access with 46.

This issues data transfer request in disk treatment SCSI controller \*tip 40, 42, 44 or vis-a-vis 46.

Directly, data transfer support software wear configuration is done as library of software \*routine, (routine it is called).

With desirable Working Example, as for file \*system, it is a client to these routine.

Consequently, when deciding that file \*system is started data transfer directly, that buffer only map calls routine which causes operation which is necessary in order to calculate checksum.

Directly when data transfer routine was done, file \*system message uses message \*system in order to send to appropriate CPU which includes disk treatment.

[0030]

Figure 7 is one example of system \*software hierarchy conversion which directly possesses data transfer.

As shown in Figure 7, as for file \*system 210, it is possible and indirectly to call message \*system 230, before calling



かつメッセージ・システムを呼出す前に直接データ転送ライブラリ 220 のルーチンを呼出すことができる。

ファイル・システム 210 は、フラグ・イネープリング直接データ転送が設定されたときに直接データ転送ライブラリ 220 を呼出す。

先に示したように、このフラグは、SETMODE 141 または BULKREAD/BULKWRITE が実行されるときにセットされる。

直接データ転送が FILE\_CLOSE 時間でイネーブルされたならば、ファイル・システムは、直接データ転送セッションを終了するために直接データ転送ルーチンを呼出す。

これは、直接データ転送ルーチンに、サーバ・ネット・コードと二つのサーバ・ネット・ID とを分解させる。

FILE\_CLOSE 時間は、これ以上のファイルが転送されないことを示す。

[0031]

本発明の別の実施例では、同じ CPU における複数の処理が同じディスクまたはテープに対して直接データ転送を行っているときに、デバイス・ハンドル・キャッシング・スキームが用いられる。

これは、CPU における同じサーバ・ネット ID を指し示すデバイス・ハンドルの複製を回避する。

デバイス・ハンドル・キャッシング・システムが用いられるときには、直接データ転送ルーチンは、直接データ転送セッションが確立されたときにデバイス・ハンドルが CPU に存在するかどうかを調べるためにチェックする。

デバイス・ハンドルが既に存在しているならば、そのデバイス・ハンドルが用いられる。

さもなければ、新しいデバイス・ハンドルがサーバ・ネット・コードを呼出すことによって生成される。

FILE\_CLOSE 時間では、デバイス・ハンドルは、除去されない。

[0032]

読取りデータ転送オペレーションに対して、チェックサム及び妥当性検査は、データ転送の終了後に行われる。

好ましい実施例では、直接データ転送起動ルーチンは、(1)チェックサム・バッファを割り当て、(2)サーバ・ネット仮想アドレス空間の中に要求プロ

message \*system , it is possible directly to call routine of data transfer library 220.

file \*system 210, when flag \* [ineeburingu ] data transfer is set directly, calls the data transfer library 220 directly.

As shown first, this flag is done, when SETMODE141 or BULKREAD/BULKWRITE is executed, set .

Directly data transfer enable was done at FILE\_CLOSE time, if is, file \*system calls data transfer routine in order directly to end data transfer session directly.

This, directly in data transfer routine , disassembles server \*network ID of server \*network \*cord and two .

FILE\_CLOSE time, fact that file above this is not transferred is shown.

[0031]

With another Working Example of this invention , when treatment of plural in the same CPU doing data transfer directly vis-a-vis same disk or the tape , it can use device \*handle \*caching \*scheme .

As for this, duplication of device \*handle which indicates same server \*network ID in CPU is evaded.

When using device \*handle \*caching \*system , directly when data transfer session is established directly, check it designates data transfer routine , as in order to inspect whether or not device \*handle exists in CPU .

device \*handle exists already, if is, it can use device \*handle .

Otherwise, it is formed by fact that new device \*handle calls the server \*network \*cord .

At FILE\_CLOSE time, as for device \*handle , it is not removed.

[0032]

Vis-a-vis reading data transfer operation , checksum and adequacy inspection are done after ending of data transfer .

With desirable Working Example , as for data transfer starting routine , it allots (1) checksum \*buffer directly, map does buffer of required processor in (2) server \*network virtual

セッサのバッファをマップし、かつ(3)ファイル・システムにリターンする。

マッピングは、サーバ・ネット・アダプタにサーバ・ネットの1次(主要)経路にわたりデータをプッシュさせるために行われる。

このマッピング・オペレーションは、要求 CPU のメモリがプロセッサのキャッシュと一貫しているようにするプロセッサ・キャッシュ・スウィープを含む。

ファイル・システムは、マップされたサーバ・ネット仮想アドレスをその要求制御領域に組込む。

次いで、関連メッセージは、適切な媒体サーバ処理(ディスク処理またはテープ処理)に送られる。

要求処理からのこのメッセージは、転送が直接データ転送であることを示す。

次いで、ディスク処理は、応答データ・バッファ及びチェックサム・バッファは、サーバ・ネット・アダプタによるアクセスに対してマップされることを知る。

ディスク処理は、(1)どのオペレーションを実行すべきか及び(2)指定されたアドレスにおける宛先 CPU(要求処理を含んでいる CPU)に読取りデータ及びチェックサムを配置すること、をそれに報告するサーバ・ネット・アダプタに指令を発行する。

サーバ・ネット・アダプタは、データ転送が終了したときにディスク処理を含んでいる CPU に割込む。

この構成は、図 4 に示されている。

【0033】

好ましい実施例では、サーバ・ネット仮想アドレス空間の中への要求プロセッサのバッファのマッピングは、直接データ転送ルーチンによって行われる。

これらのサーバ・ネット仮想アドレスは、ディスク処理へ転送される。

この転送を容易にするために、直接データ転送ルーチンは、それがディスク処理を含んでいる CPU に送る要求にサーバ・ネット仮想アドレスを組込むファイル・システムにサーバ・ネット仮想アドレスをリターンする。

データ転送が終了したときに、サーバ・ネット・アダプタは、要求が行う割込みを介してディスク処理を通知する。

address space , at same time return makes (3) file \*system .

mapping in server \*network \*adapter is done in order push to do data over primary (Principal) path of server \*network .

processor \*cache \*sweep where as for this mapping \*operation , memory of required CPU is consistent, cache of processor requires is included.

file \*system installs server \*network virtual address which map is done in required control region .

Next, related message is sent to appropriate media server treatment (disk treatment or tape treatment).

this message from request treatment shows fact that transfer is the data transfer directly.

Next, as for disk treatment, as for response data \*buffer and checksum \*buffer , you know that map it is done with server \*network \*adapter vis-a-vis access .

As for disk treatment, (1) it should execute which operation and (2) reading data and checksum are arranged in address CPU (CPU which includes request treatment) in the address which is appointed, command is issued in server \*network \*adapter which is reported to that.

server \*network \*adapter , when data transfer ends, interrupts CPU which includes disk treatment.

this configuration is shown in Figure 4 .

[0033]

With desirable Working Example , as for mapping of buffer of the required processor to in server \*network virtual address space , it is done directly with data transfer routine .

These server \*network virtual address are transferred to disk treatment.

In order to make this transfer easy, server \*network virtual address return is done in file \*system which installs server \*network virtual address in request which is sent to the CPU to which as for data transfer routine , that includes disk treatment directly.

When data transfer ends, server \*network \*adapter through interruption which request does notifies disk treatment.

要求処理が、データ転送終了したというメッセージを受け取ったときに、それは、読取り要求の終了を処理するために直接データ転送ルーチンを出す。

次いで、呼出された直接データ転送ルーチンは、要求 CPU のバッファに対する適切なチェックサムを計算するためにサーバ・ネット・コードを出す。

このチェックサム・オペレーションは、ブロッキング・オペレーションである。

従って、処理は、チェックサム・オペレーションが処理中の間、中断される。

ルーチンは、チェックサム計算が終了したときに呼出し処理をアンブロックする(ブロックしない)。

次いで、直接データ転送ルーチンは、計算されたチェックサムがサーバ・ネット・アダプタによってリターンされたチェックサムと一致することを確認する。

次いで、ルーチンは、呼出しファイル・システム・ルーチンへ成功またはチェックサム・エラー指示のいずれかをリターンする。

【0034】

図 8 は、直接データ転送を有する読取り要求経路での処理フローを示す。

段階 300 では、アプリケーションは、読取り要求に対するファイル・システムを出す。

段階 302 では、ファイル・システムは、要求セクションを準備しかつ直接データ転送ルーチンを出す。

段階 304 では、直接読取り起動ルーチンは、FLEXPOOL からのチェックサム・バッファを割り当てる。

段階 306 では、直接読取り起動ルーチンは、サーバ・ネットにわたりサーバ・ネット・アダプタへのバッファのアクセスをイネーブルするために要求 CPU のバッファをマップし、かつサーバ・ネット仮想アドレスをファイル・システムにリターンする。

段階 308 では、ファイル・システムは、ディスク処理へメッセージを送るためにメッセージ・システムを出す。

このメッセージは、要求セクションに組込まれるバッファのサーバ・ネット仮想アドレスを含む。

段階 310 では、ディスク処理は、サーバ・ネット・アダプタへワーク要求を発行する。

When receiving message that, request treatment ended, data transfer, that calls data transfer routine in order to treat end of reading request directly.

Directly data transfer routine which next, is called calls server \*network \*cord in order to calculate appropriate checksum for required CPU's buffer .

this checksum \*operation is blocking \*operation .

Therefore, treatment is discontinued while checksum \*operation is in midst of treating.

When checksum calculation ends, it calls routine and, treats a block (block it does not do).

Next, directly as for data transfer routine , checksum which was calculated being server \*network \*adapter , verifies that it agrees with checksum which return is done.

Next, it calls routine and, return does any of success or checksum \*error display to file \*system \*routine .

[0034]

Figure 8 shows process flow with reading required path which directly possesses data transfer .

With step 300, as for application , file \*system for reading request is called.

With step 302, as for file \*system , required section only preparation data transfer routine is called directly.

With step 304, as for reading starting routine , checksum \*buffer from the FLEXPOOL is allotted directly.

With step 306, directly as for reading starting routine , the required CPU's buffer map is done in order to enable to do access of buffer to server \*network \*adapter over server \*network , at same time server \*network virtual address the return is designated as file \*system .

With step 308, as for file \*system , message \*system is called in order to send message to disk treatment.

this message includes server \*network virtual address of buffer which is installed in the required section .

With step 310, as for disk treatment, work request is issued to the server \*network \*adapter .

このワーク要求は、読取りデータ及びチェックサムを、要求処理を含んでいるCPUに配置するためのものである。

段階 312 では、サーバ・ネット・アダプタは、データ転送を実行しそして終了通知でディスク処理に割込む。

段階 314 では、ディスク処理は、要求処理のメッセージに回答する。

[0035]

図 9 は、直接データ転送を伴う読取り応答経路に対する処理フローを示す。

段階 320 では、ディスク処理の応答は、要求処理のメッセージをキューにさせ、かつ要求処理が起動される。

段階 322 では、要求処理が起動しかつチェックサムを確認する。

段階 324 では、直接読取り終了ルーチンは、DMA エンジンに対するチェックサム計算をキューするためにサーバ・ネット・コードを呼出す。

次いで、呼出し処理を中断する。

DMA エンジンは、次いで、チェックサム計算を実行する。

段階 326 では、DMA エンジン・チェックサム計算終了割込みは、サーバ・ネット・コードに要求処理をアンブロックさせる。

段階 328 では、直接データ転送ルーチンは、サーバ・ネット・アダプタによってデポジット(配置)されたチェックサムで計算されたチェックサムを確認する。

段階 330 では、直接データ転送ルーチンは、FLEXPOOL にチェックサム・バッファをリターンしかつファイル・システムに対するチェックサム比較の成功または失敗をリターンする。

段階 332 では、好ましい実施例では、ファイル・システムは、比較が成功であれば、要求処理へリターンするか、または、チェックサム・エラーが発生したならば、直接データ転送なしで再度試みる。

[0036]

書き込みデータ転送オペレーションに対して、チェックサムは、データが転送される前に計算される。

ファイル・システムは、直接書き込み起動ルーチンを呼出しかつ要求 CPU のバッファのアドレスを供給する。

this work request reading data and checksum , is something in order to arrange in CPU which includes request treatment.

With step 312, as for server \*network \*adapter , it executes data transfer and and interrupts disk treatment with end notification.

With step 314, as for disk treatment, you respond to message of request treatment.

[0035]

Figure 9 shows process flow for reading response path which directly accompanies data transfer .

With step 320, as for response of disk treatment, at same time request treatment is started with message of request treatment as queue .

With step 322, request treatment only starting verifies checksum .

With step 324, directly as for reading end routine , server \*network \*cord is called in order queue to do checksum calculation for DMA engine .

Next, it calls and discontinues treatment.

DMA engine , next, executes checksum calculation.

With step 326, as for DMA engine \*checksum calculation end interruption, request treatment and block is done in server \*network \*cord .

With step 328, directly as for data transfer routine , deposit checksum which was calculated with checksum (Arrangement) is verified with server \*network \*adapter .

With step 330, directly as for data transfer routine , checksum \*buffer only return its succeeds checksum comparison for file \*system in FLEXPOOL or it fails return .

If with step 332, with desirable Working Example , as for file \*system , comparison is success, to request treatment return it does, the checksum \*error occurred, directly with data transfer none for second time it tries.

[0036]

Vis-a-vis entry data transfer operation , checksum is calculated before data is transferred.

file \*system direct entry starting routine supplies address of the buffer of call and required CPU .

供給する。

直接データ転送ルーチンは、チェックサム・バッファをまず割り当てかつ関連バッファ(要求データ及びチェックサム・バッファ)をサーバ・ネット仮想アドレス空間の中にマップする。

書込みオペレーションに対して、バッファは、ディスク及びそのミラーの二つの経路によりアクセスに対してサーバ・ネット仮想アドレス空間の中にマップされる。

これがミラーされた書込みであれば、両方の関連サーバ・ネット・アダプタは、これらのバッファをアクセスする。

ミラーされた書込みは、好ましい実施例で用いられる。

[0037]

次いで、直接データ転送ルーチンは、バッファのチェックサムを計算するためにサーバ・ネット・コードを呼出す。

サーバ・ネット・コードは、要求CPUのDMAエンジンにチェックサム計算を次いでキューしかつチェックサム計算の終了まで呼出し処理をブロックする。

DMAエンジンは、指定したチェックサムを計算しかつ合成チェックサムをチェックサム・バッファに配置する。

チェックサム計算に対する終了割り込みは、サーバ・ネット・コードによって次いでフィールドされる。

終了割り込みは、その時に要求処理をアンブロックする。

直接データ転送ルーチンは、ファイル・システムに4つのサーバ・ネット仮想アドレスをリターンすることによって処理を再開する。

これらのサーバ・ネット仮想アドレスの二つは、要求CPUのバッファに対応する。

他の二つのサーバ・ネット仮想アドレスは、チェックサム・バッファに対応する。

ファイル・システム書込みルーチンは、これらの4つのサーバ・ネット仮想アドレスを、そのメッセージの要求制御セクションの中に組みそして関連メッセージをディスク処理へ送るためにメッセージ・システムを呼出す。

好ましい実施例では、データ汚染を回避するために、要求処理バッファは、ディスク読み取りが実行されるときに一つのサーバ・ネット・アダプタに

Directly data transfer routine checksum \*buffer first map designates allotment and related buffer (Required data and checksum \*buffer) as in server \*network virtual address space .

Vis-a-vis entry operation , buffer map makes in server \*network virtual address space with path of two of disk and its mirror vis-a-vis access .

If this is entry which mirror is done, related server \*network \*adapter of the both access does these buffer .

Entry seeing which mirror is done is used with desirable Working Example .

[0037]

Next, directly data transfer routine calls server \*network \*cord in order to calculate the checksum of buffer .

server \*network \*cord only queue calls checksum calculation to end of checksum calculation next in DMA engine of required CPU and treats the block .

DMA engine checksum which is appointed only calculation arranges the synthetic checksum in checksum \*buffer .

End interruption for checksum calculation field is done next with server \*network \*cord .

End interruption and block does request treatment that time.

Directly data transfer routine reopens treatment by fact that server \*network virtual address of 4 return is done in file \*system .

two of these server \*network virtual address corresponds to buffer of required CPU .

server \*network virtual address of other two corresponds to checksum \*buffer .

file \*system entry routine server \*network virtual address of these 4, calls message \*system in order to send insertion and related message to disk treatment in therequired control section of message .

With desirable Working Example , in order to evade data pollution, as forrequired treatment buffer , when disk reading is executed, the map it is done to just server \*network

だけマップされる。

[0038]

ディスク処理は、この直接データ転送書き込み要求の受け取りにより、SCSIコントローラへ適切なワーク要求を送る。

このワーク要求は、データ及びチェックサム・バッファが要求処理を含んでいる CPU から来ることを指定するが、サーバ・ネット・アダプタからの要求終了割り込みは、ディスク処理を含んでいる CPU に行くべきであることを指定する。

次いで、サーバ・ネット・アダプタは、要求処理の CPU から書き込みオペレーションに対するデータをプルする。

次いで、データは、物理的媒体に書き込まれ、そしてディスク処理は、終了により割り込みを介して通知される。

ディスク処理は、1 次及びミラー・ディスクの半分の両方からの終了割り込みを待つ。

両方のディスクがオペレーションを終了した後、ディスク処理は、要求処理のメッセージに応答する。

[0039]

ディスク処理によって送られた応答が要求処理の CPU に到着したとき、CPU は、要求処理を起動する。

次いで、ファイル・システムは、書き込み終了を処理するために直接データ転送ルーチンと呼出す。

この場合には、直接データ転送ルーチンは、サーバ・ネット仮想アドレス空間からのバッファをアンマップ(unmaps)し、チェックサム・バッファの割り当てを解除し、かつファイル・システムにリターンする。

次いで、ファイル・システムは、そのユーザに書き込み要求の終了を通知する。

図 10 は、直接データ転送を伴う書き込み要求経路に対する処理フローを示す。

段階 350 では、アプリケーションは、書き込みオペレーションに対するファイル・システムを呼出す。

段階 352 では、ファイル・システムは、直接データ転送書き込み起動ルーチンと呼出す。

段階 354 では、直接データ転送書き込み起動ルーチンは、FLEXPOOL からのチェックサム・バッファを割り当てる。

\*adapter of one .

[0038]

As for disk treatment, appropriate work request is sent to SCSI controller this due to receipt of data transfer entry request directly.

this work request appoints that it comes from CPU to which data and checksum \*buffer include request treatment, but, it appoints that therequest end interruption from server \*network \*adapter should go to CPU whichincludes disk treatment.

Next, server \*network \*adapter [puru ] data from CPU of request treatment forentry operation .

Next, data is written by physical media , and disk treatment isnotified through interruption, with end.

disk treatment waits for end interruption from both of the half of primary and mirror \*disk .

After disk of both ends operation , as for disk treatment,you respond to message of request treatment.

[0039]

When response which is sent in disk treatment arrives at the CPU of request treatment, CPU starts request treatment.

Next, file \*system calls data transfer routine in order to treat entry end directly.

In case of this , data transfer routine ane map (unmaps ) does buffer from the server \*network virtual address space directly, cancels allotment of checksum \*buffer , at same time return makes file \*system .

Next, file \*system notifies end of entry request to user .

Figure 10 shows process flow for entry required path which directlyaccompanies data transfer .

With step 350, as for application , file \*system for entry operation iscalled.

With step 352, as for file \*system , data transfer entry starting routine iscalled directly.

With step 354, as for data transfer entry starting routine , checksum \*buffer from FLEXPOOL is allotted directly.

段階 356 では、直接データ転送書込み起動ルーチンは、二つのコントローラ経路のよるアクセスに対する要求 CPU のバッファ及びチェックサム・バッファをマップする。

次いで、ルーチンは、チェックサム計算を行うためにサーバ・ネット・コードを呼出す。

段階 358 では、サーバ・ネット・コードは、チェックサム計算を DMA エンジンにキューしかつ要求処理を中断する。

段階 360 では、終了割り込みは、サーバ・ネット・コードに要求処理を再開させかつ直接書込み起動ルーチンにリターンさせる。

段階 362 では、直接データ転送書込み起動は、要求及びチェックサム・バッファのサーバ・ネット仮想アドレスをファイル・システムにリターンする。

段階 364 では、ファイル・システムは、これらのサーバ・ネット仮想アドレスをその要求制御領域に組み込みかつこのメッセージをディスク処理へ送る。

[0040]

図 11 は、直接データ転送を伴う書込み応答経路に対する処理フローを示す。

段階 370 では、コードは、要求処理を起動する。

段階 372 では、ファイル・システムは、直接データ転送書込み終了ルーチンを呼出す。

段階 374 では、この直接データ転送ルーチンは、サーバ・ネット仮想アドレス空間からのバッファをアンマップし、チェックサム・バッファの割り当てを解除してリターンする。

段階 376 では、ファイル・システムは、アプリケーションに書込み終了通知をリターンする。

好ましい実施例では、チェックサム・バッファに対する空間は、直接データ転送ライブラリ・ルーチンによって割り当てられる。

この空間は、FLEXPOOL から割り当てられる。

代替実施例では、ファイル・システムは、チェックサム・バッファに対する空間を割り当てる。

チェックサム・バッファ・サイズは、要求 CPU のバッファ・サイズ及びセクタ/ブロック・サイズに依存する。

[0041]

好ましい実施例では、直接データ転送オペレーションを起動するファイル・システム・クライアント

With step 356, buffer and checksum \*buffer of required CPU for the access to which data transfer entry starting routine depends controller path of the two directly are done map .

Next, routine calls server \*network \*cord in order to calculate checksum .

With step 358, as for server \*network \*cord , checksum calculation only queue request treatment is discontinued in DMA engine .

With step 360, as for end interruption, it makes requesttreatment server \*network \*cord reopen and and return makes direct entrystarting routine .

With step 362, directly as for data transfer entry starting, server \*network virtual address ofrequest and checksum \*buffer return is designated as file \*system .

With step 364, as for file \*system , these server \*network virtual address insertion and this message are sent to disk treatment in required control region .

[0040]

Figure 11 shows process flow for entry response path which directlyaccompanies data transfer .

With step 370, as for cord , request treatment is started.

With step 372, as for file \*system , data transfer entry end routine is calleddirectly.

With step 374, this as for data transfer routine , ane map it does the buffer from server \*network virtual address space directly, cancels allotment of checksum \*buffer and return does.

With step 376, as for file \*system , entry end notification return isdone in application .

With desirable Working Example , as for space for checksum \*buffer , it isallotted directly with data transfer library \*routine .

this space is allotted from FLEXPOOL.

With alternate embodiment , as for file \*system , space for checksum \*buffer is allotted.

checksum \*buffer \*size depends on buffer \*size and sector /block \*size of required CPU .

[0041]

With desirable Working Example , as for file \*system \*client which directly starts the data transfer operation , in addition, it

は、また、そのオペレーションを取り消すこともできる。

この取消しは、データ転送の終りでまたは顕著なデータ転送要求が存在する間に行うことができる。

この方法で直接データ転送オペレーションが取り消されるとき、ファイル・システムは、取消しを実行するために直接データ転送ルーチンと呼出す。

このルーチンは、サーバ・ネット仮想アドレス空間からのバッファをアンマップすることに対しかつ関連チェックサム・バッファをそれらのプールにリターンすることに対して責任がある。

ファイル・システムは、次いで、取消し通知を適切なディスク処理へ送る。

これら二つのオペレーションが終了した後、直接データ転送オペレーションが取り消されと考えられる。

直接データ転送ルーチンがバッファをアンマップするとき、サーバ・ネット・アダプタは、それがこのバッファをアクセスすることを試みるならばエラーに遭遇する。

これらのエラーは、サーバ・ネット・アダプタ及び要求処理を含んでいる CPU の両方に報告される。

好ましい実施例では、サーバ・ネット・アダプタは、第 1 のエラーに遭遇した後バッファをアクセスすることを停止する。

[0042]

好ましい実施例では、システムにおける全ての CPUs のサーバ・ネット ID は、ファームウェアで登録される。

この登録は、ファームウェアにおける“セット・サーバ・ネット・パラメータ”メールボックス指令の形である。

このメールボックス指令に応じて、ファームウェアは、8ビット・サーバ・ネット・ハンドルをリターンする。

このハンドルは、ファームウェアに対するホスト・プロセッサを識別するために SCSI モジュール・ドライバによって用いられる。

このサーバ・ネット IDs の登録は、要求プロセッサ(要求処理を含むプロセッサ)と記憶装置(ディスクを含む装置)との間で直接データ転送を許容する。

is possible also to cancel operation .

As for this cancellation, or while marked data transfer request exists, it is possible with end of data transfer to do.

When data transfer operation is cancelled directly with this method , file \*system calls data transfer routine in order to execute cancellation directly.

this routine is a responsibility confronting and related checksum \*buffer vis-a-vis return making those pool in ane map doing buffer from server \*network virtual address space .

Next, file \*system , notice of cancellation is sent to appropriate disk treatment.

After operation of these two ends, you can think data transfer operation cancellation directly.

Directly when data transfer routine ane map does buffer , server \*network \*adapter tries the fact that that access does this buffer , if is, you encounter to the error .

These error are reported to both of CPU which includes the server \*network \*adapter and request treatment.

With desirable Working Example , as for server \*network \*adapter , after encountering to the first error , fact that buffer is done access is stopped.

[0042]

With desirable Working Example , server \*network ID of all CPUs in system is registered with firmware .

As for this register, it is a shape of "set \*server \*network \*parameter \*mail box command in firmware .

According to this mail box command , firmware return does 8 bit \*server \*network \*handle .

this handle in order to identify host \*processor for firmware is used with SCSI module \*driver .

Register of this server \*network IDs required processor (processor which includes request treatment) with allows data transfer directly between storage device (device which includes disk ) .



上記で説明した例は、要求処理及びディスク処理が異なる CPUs に配置される場合だけを考慮した。

これら二つの処理が同じ CPU に配置されるならば、直接データ転送は、まだ用いることができる。

この例で直接データ転送が用いられるときに、要求処理とディスク処理との間のバッファの CPU コピーは、まだ回避される。

従って、また、直接データ転送は、要求処理及びディスク処理の両方が同じ CPU に配置されるときにも望ましい。

[0043]

上記の例では、ディスクは、記憶デバイスとして用いられたが、直接データ転送に対してあらゆる記憶デバイスを用いることができる。

例えば、直接データ転送は、不必要なデータ・コピーを回避するためにテープ I/O に用いることができる。

その I/O がチェックサム計算を含まないならば、直接データ転送は、ディスクの場合よりも簡単である。

テープ処理が直接データ転送も支持するならば、バックアップ及び復元施設は、この特徴を利用することができる。

例えば、ディスク処理の場合、テープ処理及びバックアップ/復元処理は、3つの異なる CPUs に全て分散され、直接データ転送は、二つの不必要なバッファ・コピーを回避しうる。

好ましい実施例では、次に示すソフトウェア・コードの抜粋は、直接データ転送ライブラリにおけるインターフェイス・ルーチンを供給する。

第 1 の抜粋は、直接データ転送セッション開始ルーチン(このパラグラフの真下を参照)に対するものである。

このルーチンは、それが直接データ転送要求を支持することができるディスク処理からの指示をそれが受け取るときにファイル・システムによって呼出される。

ディスク処理は、この指示を有する SETMODE 141/bulk データ転送要求に応答する。

ディスク処理からの応答は、また、サーバ・ネットワーク IDs、パケットサイズ型、等のような、直接データ転送ルーチンによって用いられる情報を含む。

この情報は、ファイル・システムによって直接デ

When request treatment and disk treatment are arranged in different CPUs only, it considered example which is explained at description above.

If treatment of these two is arranged in same CPU, still you can use data transfer, directly.

When using data transfer directly with this example, CPU copy of the buffer during request treatment and disk treatment is still evaded.

Therefore, in addition, directly data transfer, when both of request treatment and disk treatment is arranged in same CPU even, is desirable.

[0043]

With above-mentioned example, as for disk, it was used as the storage device all storage device can be used, but directly vis-a-vis data transfer.

for example directly you can use data transfer, for tape I/O in order to evade unnecessary data \*copy.

I/O does not include checksum calculation, if is, directly the data transfer is simple in comparison with when it is a disk.

If tape treatment is also data transfer supports directly, backup and restoration infrastructure can utilize this feature.

In case of for example disk treatment, tape treatment and backup / restoration treatment all are dispersed by 3 different CPUs, data transfer can evade the unnecessary buffer \*copy of two directly.

With desirable Working Example, as for excerpt of software \*cord which is shown next, interface \*routine directly in data transfer library is supplied.

first excerpt is something directly for data transfer session start routine (You refer to directly below of this paragraph).

this routine when that receives display from disk treatment where that directly can support data transfer request, is called with file \*system.

As for disk treatment, you respond to SETMODE 141/bulk \*data transfer request which possesses the this display.

As for response from disk treatment, in addition, server \*network IDs, [pakettaiza] type, the data which or other, directly is used with data transfer routine is included.

this data with file \*system pass makes directly data transfer

一タ転送セッション開始ルーチンにパスされる。 session start routine .

[0044] [0044]

[表 1] [Table 1 ]

```

/*
.. Returns 0 upon success
.. an error code upon failure
.. iop_cookie is an input pointer to the area that the IOP returned to the
.. File System to pass on to the Direct IO routines.
.. The File System should not care
.. about the internal format of this area. The format of this area would be
.. an agreement between the IOPs and the Direct IO routines. This area
.. will contain the Tnet IDs of the controller, the type of the packetizer
.. in the controller, etc.
..
.. directio_cookie is a pointer to an area where directio can store some
.. of its context for the directio session. Aside from allocating storage for
.. this area, the File System should not care about the format of this area.
..
.. The size of the cookie areas is TBD.
*/
int DirectIO_Session_Start(void *iop_cookie, void *directio_cookie);
{
    look at iop cookie and see how many tnet ids are in it.
    call tser_dev_install to install these tnet ids. Ask tser_dev_install not to
    worry about allocating interrupt and barrier AVTs.
    call tser_dev_set_packetizer to set the packetizer to the type specified
    in the iop_cookie.
    call tser_dev_set_tnetid to set the tnet id in the installed device handle.
    store the returned device handles in the directio cookie area.
    allocate a Tib from FLEXPOOL and store the TIB address in the
    directio_cookie area.
}

```

直接データ転送セッション終了ルーチンは、FILE\_CLOSE 時間でまたは SETMODE 141 が大きいデータ転送を不能にするために実行されるときにファイル・システムによって呼出される。

Directly data transfer session end routine , when or being executed in order to make data transfer where SETMODE141 is large impossible at FILE\_CLOSE time, is called with file \*system .

[0045] [0045]

[表 2] [Table 2 ]

```
.. Returns 0 upon success
.. an error code upon failure.
..
.. directio_cookie is a pointer to the area that was initialized during the
.. directio_session_start call.
./
int DirectIO_Session_End(void *directio_cookie)
(
    call tser_dev_remove to uninstall the Tnet IDs.
    deallocate the TIB buffer by returning it to the FLEXPOOL.
)
```

直接読取り開始ルーチンは、直接データ転送読取り要求を起動する前にファイル・システムによって呼出される。

【0046】

【表 3】

Directly reading start routine before directly starting data transfer reading request, is called with file \*system .

[0046]

[Table 3 ]

```

/*
** Returns 0 upon success
**      an error code upon failure.
**
** directio_cookie is a pointer to the direct io session information that was
**      initialized during directio session start.
** buffer is a pointer to the buffer into which data should be read.
** buffer_size is the size of the above buffer.
** tnet_vaddrs is a pointer to an array of two 32 bit integers. Two tnet
**      virtual addresses will be deposited there by Direct_Read_Start.
**      These two tnet vaddrs then need to be copied to the Request Control
**      and sent on to the IOP.
** xsum_buffer is a pointer to a 32 bit integer location. Direct_Read_Start
**      deposits the address of the xsum buffer it allocated in this location.
**      This address needs to be passed to Direct_Read_End or
**      Direct_Read_Abort.
*/
int Direct_Read_Start(void *directio_cookie, void *buffer, int buffer_size,
                    void *tnet_vaddrs, void *xsum_buffer)
{
    *xsum_buffer = allocate the checksum buffer from the Flex Pool.
    map the user buffer and the checksum buffer for write access by the
        device handle stored in the directio_cookie.
    deposit the two tnet virtual addresses returned by the map routine into the
        area pointed at by tnet_vaddrs.
}

```

直接読取り終了ルーチンは、それがディスク処理から直接データ転送読取り要求に対する応答を受け取るときにファイル・システムによって呼出される。

ディスク処理応答が成功を示すならば、直接読取り終了ルーチンは、チェックサム妥当性検査を行う。

全ての場合に、直接読取り終了ルーチンは、バッファをアンマップしかつチェックサム・バッファの割り当てを解除する。

[0047]

[表 4]

Directly reading end routine when receiving response that from disk treatment directly for data transfer reading request, is called with the file \*system .

disk treatment response shows success, if is, reading end routine inspects directly checksum adequacy .

In case of all , reading end routine buffer only ane map cancels allotment of checksum \*buffer directly.

[0047]

[Table 4 ]

```

/*
-- Returns 0 upon success
.. an error code upon failure. One such failure could be checksum mismatch.
..
.. tnet_vaddrs is a pointer to the two tnet virtual addresses that were created
.. during the call to Direct_read_start.
.. iop_returned_status is the status code that the IOP returned in its response to
.. the read request sent by the File System.
.. xsum_buffer is the 32 bit address of the checksum buffer that Direct_Read_Start
.. allocated.
*/
int Direct_Read_End(void *directio_cookie, void *tnet_vaddrs, int iop_returned_status,
                   void xsum_buffer)
{
    unmap the two tnet virtual addresses listed in the tnet_vaddrs array.

    if (iop_returned_status == success)
    {
        use the TIB from the directio cookie area.
        call tser_transfer to calculate checksum.
        compare device returned checksum to calculated checksum.
        set return value.
    }
    return xsum_buffer to FLEXPOOL.
    return return_value
}

```

直接書き込み開始ルーチンは、直接データ転送書き込み要求を起動する前にファイル・システムによって呼出される。

Direct entry start routine before directly starting data transfer entryrequest, is called with file \*system .

【0048】

【0048】

【表 5】

【Table 5】

```

/*
.. Returns 0 upon success.
.. otherwise an error code is returned.
..
.. directio_cookie is a pointer to the area of memory that belongs to Direct IO
.. that is allocated by the File System.
..
.. buffer is a pointer to the user buffer that is being written to the media.
.. buffer_size is the size of the above buffer.
.. tnet_vaddrs is a pointer to an array of four 32 bit integers. The array needs to
.. be allocated by the file system. Direct IO routines will deposit four Tnet
.. virtual addresses into this array - two for the xsum buffer and two for the user
.. buffer. The FS needs to copy these four into the request control it sends to
.. the IOP.
..
.. xsum_buffer is a pointer to a 32 bit location where Direct_Write_Start will
.. deposit the address of the allocated xsum buffer. This address needs
.. to be passed to Direct_Write_End or Direct_Write_Abort.
*/
int Direct_Write_Start(void *directio_cookie, void *buffer, int buffer_size,
                      void *tnet_vaddrs, void *xsum_buffer)
{
    *xsum_buffer = allocate a xsum buffer from the FLEXPOOL.
    call the Tnet millicode map routines to map the xsum and user buffers for
    access by the disk and its mirror.
    deposit the Tnet virtual addresses returned above into tnet_vaddrs array.
    use the TIB whose address we stored in the directio_cookie.
    call tser_transfer to calculate checksum of buffer.
}

```

直接書き込み終了ルーチンは、直接データ転送書き込み要求への応答がディスク処理から受け取られるときにファイル・システムによって呼出される。

[0049]

[表 6]

Direct entry end routine , when directly response to data transfer entryrequest is received from disk treatment, is called with file \*system .

[0049]

[Table 6 ]

```

.. Returns 0 upon success.
.. an error code is returned otherwise.
..
.. directio_cookie is a pointer to the area of File System memory where directio
.. session information is stored.
.. tnet_vaddrs is a pointer to an array of four 32 bit ints. This is the same as the
.. array which was passed into direct_write_start.
~/
int Direct_Write_End(void *directio_cookie, void *tnet_vaddrs, void *xsum_buffer)
{
    call tnet millicode to unmap the four tnet virtual addresses listed in
    tnet_vaddrs.
    deallocate xsum_buffer.
}

```

直接読み取りアボートルーチンは、それがディスク処理に送った直接データ転送読み取り要求をそれが取り消した後にファイル・システムによって呼出される。

Directly reading [abooto] \* routine after directly that cancels data transfer which that sends to disk treatment reading request is called with the file \*system .

ディスク処理は、取消しを知らされ、そして要求をクリーンにするために直接データ転送ルーチンが呼出される。

disk treatment informs about cancellation, and data transfer routine is called in order to designate request as clean directly.

【0050】

【0050】

【表 7】

【Table 7】

```

~/
.. Returns 0 upon success.
.. an error code is returned otherwise.
.. directio_cookie is a pointer to the area of the File System memory where directio
.. session information is stored.
.. tnet_vaddrs is a pointer to an array of two 32 bit ints. This is the same as the
.. array which was passed into direct_read_start.
.. xsum_buffer is the 32 bit address of the checksum buffer that Direct_Read_Start
.. allocated.
~/
int Direct_Read_Abort(void *directio_cookie, void *tnet_vaddrs, void *xsum_buffer)
{
    call tnet millicode to unmap the two tnet virtual addresses listed in
    tnet_vaddrs.
    deallocate xsum_buffer.
}

```

直接書き込みアボート・ルーチンは、それがディスク処理に送った直接データ転送書き込みメッセージをそれが取り消した後にファイル・システムによって呼出される。

ディスク処理は、取消しについて知らされ、そして要求をクリーンにするために直接データ転送ルーチンが呼出される。

[0051]

[表 8]

```

/*
** Returns 0 upon success.
** an error code is returned otherwise.
**
** directio_cookie is a pointer to the area of File System memory where directio
** session information is stored.
** tnet_vaddrs is a pointer to an array of four 32 bit ints. This is the same as the
** array which was passed into direct_write_start.
** xsum_buffer is the 32 bit address of the checksum buffer allocated by
** Direct_Write_Start.
*/
int Direct_Write_Abort(void *directio_cookie, void *tnet_vaddrs, void *xsum_buffer)
{
    call tnet millicode to unmap the four tnet virtual addresses listed in
    tnet_vaddrs.
    deallocate xsum_buffer.
}

```

上記したように、チェックサムは、各 CPU 内のブロック転送エンジンを用いて計算される。

それゆえに、チェックサムは、CPU の介入なしで、非同期で計算することができる。

全ての CPU のソフトウェア・サブ・システムは、チェックサム計算を実行することに対して責任がある。

代替実施例では、チェックサム計算は、ファイル・システム・コードで実行することができる。

この例では、サーバ・ネット・コードが非同期インターフェイスだけを供給するので、割り込みハンドラは、チェックサム計算の終了割り込みを処理するために用いられる。

別の実施例では、サーバ・ネット・コードは、同期インターフェイスを供給する。

この構成では、チェックサムは、割り込みハンドラ

Direct entry [abooto ] \* routine after directly that cancels data transfer which that sends to disk treatment entry seeing message is called with file \*system .

disk treatment informs concerning cancellation, and data transfer routine is called in order to designate request as clean directly.

[0051]

[Table 8 ]

As inscribed, checksum is calculated making use of block transfer engine inside each CPU .

Consequently, with intervention none of CPU , it can calculate checksum , with asynchronous .

software \*sub -system of all CPU is a responsibility vis-a-vis executing checksum calculation.

With alternate embodiment , it can execute checksum calculation, with file \*system \*cord .

Because with this example, server \*network \*cord supplies just asynchronous interface , it interrupts and handler is used in order to treat end interruption of checksum calculation.

With another Working Example , as for server \*network \*cord , synchronization interface is supplied.

With this configuration , it interrupts checksum and, it can



なしでファイル・システム・レイヤにおいて実行することができる。

[0052]

本発明の全てかつ完全な開示がなされると同時に、種々の代替及び変更が、特許請求の範囲の範囲から逸脱することなく本発明の種々の態様に対してなされうということが当業者に明らかになるであろう。

[0053]

[発明の効果]

本発明のデータ処理システムは、データを転送するデータ処理システムであって、要求 CPU を含んでいる複数の中央処理装置 (CPUs); CPUs の一つが記憶装置へのアクセスを制御する少なくとも一つの記憶装置; 複数の CPUs と記憶装置とを相互接続するネットワーク; 要求 CPU による記憶装置の直接アクセスを供給し、要求 CPU のバッファに対する仮想メモリ・アドレスを生成し、かつ記憶装置アクセス要求と共に仮想メモリ・アドレスを、記憶装置へのアクセスを制御している CPUs の一つに供給する要求 CPU における手段、ワーク要求に回答しておりかつネットワークを通して CPUs の一つと直接インターフェイスしている記憶装置に仮想メモリ・アドレスを含んでいるワーク要求を送る CPUs の一つにおける手段、を含んでいるアクセス手段; を備え、データは、要求 CPU と記憶装置との間で直接転送されるので、CPUs と記憶ディスクとの間のデータの転送中に類似するデータ・コピーを除去することができる。

[図面の簡単な説明]

[図1]

記憶装置アーキテクチャの一例を示す図である。

[図2]

ディスク転送の一例を示す図である。

[図3]

コントローラが“プッシング”しているときに発生するデータの転送を示す図である。

[図4]

ディスク読取りに対する直接データ転送の一例を示す図である。

[図5]

ディスク書き込みに対する直接データ転送の一例

execute with the handler none in file \*system \*layer .

[0052]

When you can do all and complete disclosure of this invention , without deviating from category of Claims in person skilled in the art it becomes clear simultaneously, to be possible be able to do various substitution and modification, vis-a-vis various embodiment of this invention .

[0053]

[Effects of the Invention ]

As for data processing system of this invention , with data processing system which transfers the data , central processing unit of plural which includes required CPU (CPUs ); storage device to supply access directly with CPUs of storage device ; plural of at least one where one of CPUs controls access to storage device and network ; required CPU which interconnect does storage device , the hypothetical memory \*address for buffer of required CPU only formation with storage device access request hypothetical memory \*address , We to have responded to means . work request in required CPU which is supplied to one of CPUs which controls access to the storage device to have access means ; which includes means . in one of the CPUs which sends work request which includes hypothetical memory \*address in storage device which interface has been made one of the CPUs directly and through network , as for data , Because it is transferred directly required CPU and between the storage device , CPUs and data \*copy which resembles while transferring the data between storage disc can be removed.

[Brief Explanation of the Drawing (s )]

[Figure 1 ]

It is a figure which shows one example of storage device architecture .

[Figure 2 ]

It is a figure which shows one example of disk transfer.

[Figure 3 ]

When controller " [pushing ] " having done , it is a figure which shows the transfer of data which occurs.

[Figure 4 ]

It is a figure which directly shows one example of data transfer for disk reading.

[Figure 5 ]

It is a figure which directly shows one example of data

を示す図である。

【図6】

どのように二つのサーバ・ネット ID がディスク及びそのミラー・ディスクへの二つの経路に対して用いられるかを示す図である。

【図7】

直接データ転送を伴うシステム・ソフトウェア階層化の一例である。

【図8】

直接データ転送を伴う読取り要求経路を有する処理フローを示す図である。

【図9】

直接データ転送を伴う読取り応答経路に対する処理フローを示す図である。

【図10】

直接データ転送を伴う書込み要求経路に対する処理フローを示す図である。

【図11】

直接データ転送を伴う書込み応答経路に対する処理フローを示す図である。

【符号の説明】

- 10
- サーバ・ネット雲
- 105
- ディスク
- 120
- SCSI チェイン
- 162
- バッファ
- 170
- データ
- 172
- サーバ・ネット仮想アドレス
- 174
- ソース・ノード
- 176
- 宛先ノード

transfer for disk entry.

[Figure 6]

Which way, it is a figure which shows whether server \*network ID of two it is used vis-a-vis path of two to disk and its mirror \*disk .

[Figure 7]

It is a one example of system \*software hierarchy conversion which directly accompanies the data transfer .

[Figure 8]

It is a figure which shows process flow which possesses readingrequired path which directly accompanies data transfer .

[Figure 9]

It is a figure which shows process flow for reading response path which directly accompanies data transfer .

[Figure 10]

It is a figure which shows process flow for entry required path which directly accompanies data transfer .

[Figure 11]

It is a figure which shows process flow for entry response path which directly accompanies data transfer .

[Explanation of Symbols in Drawings]

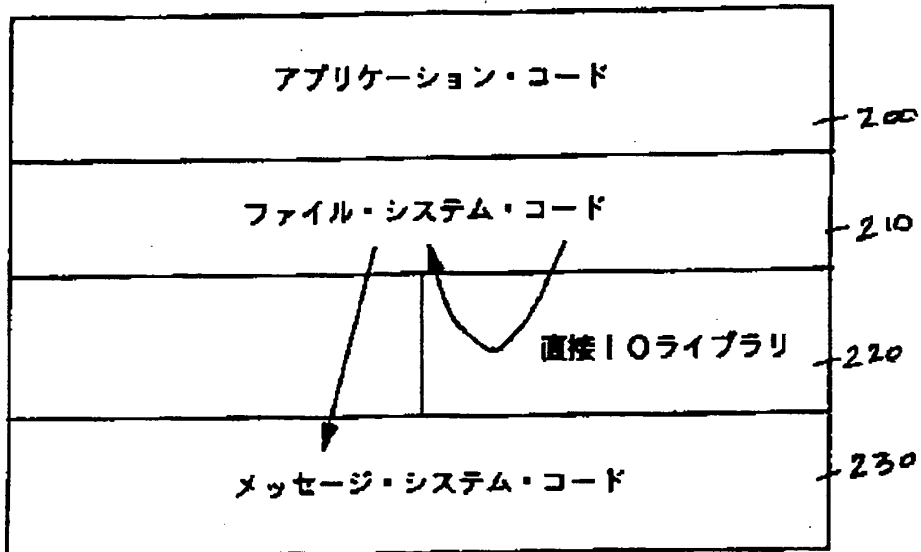
- 10
- server \*network cloud
- 105
- disk
- 120
- SCSI chain
- 162
- buffer
- 170
- data
- 172
- server \*network virtual address
- 174
- source \*node
- 176
- addressee node

26	26
CPU3	CPU 3
30	30
サーバ・ネット・アダプタ	server *network *adapter
34	34
I/O パケッタイザ	I/O [pakettaiza ]
40	40
SCSI チップ	SCSI tip

Drawings

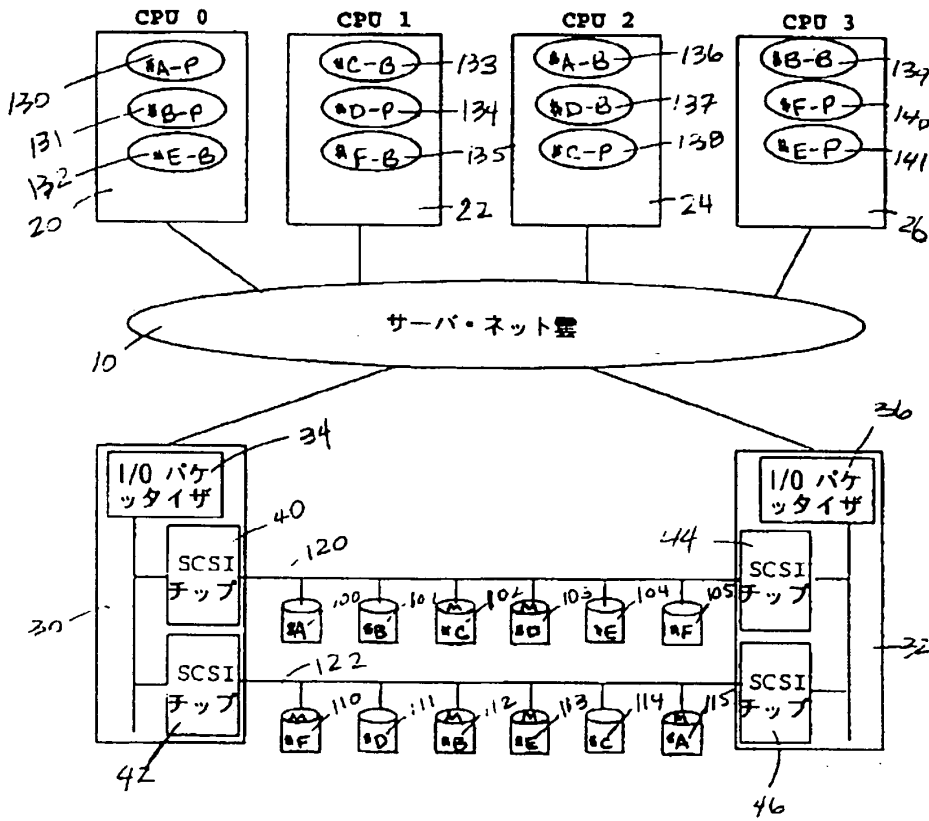
【図7】

[Figure 7]



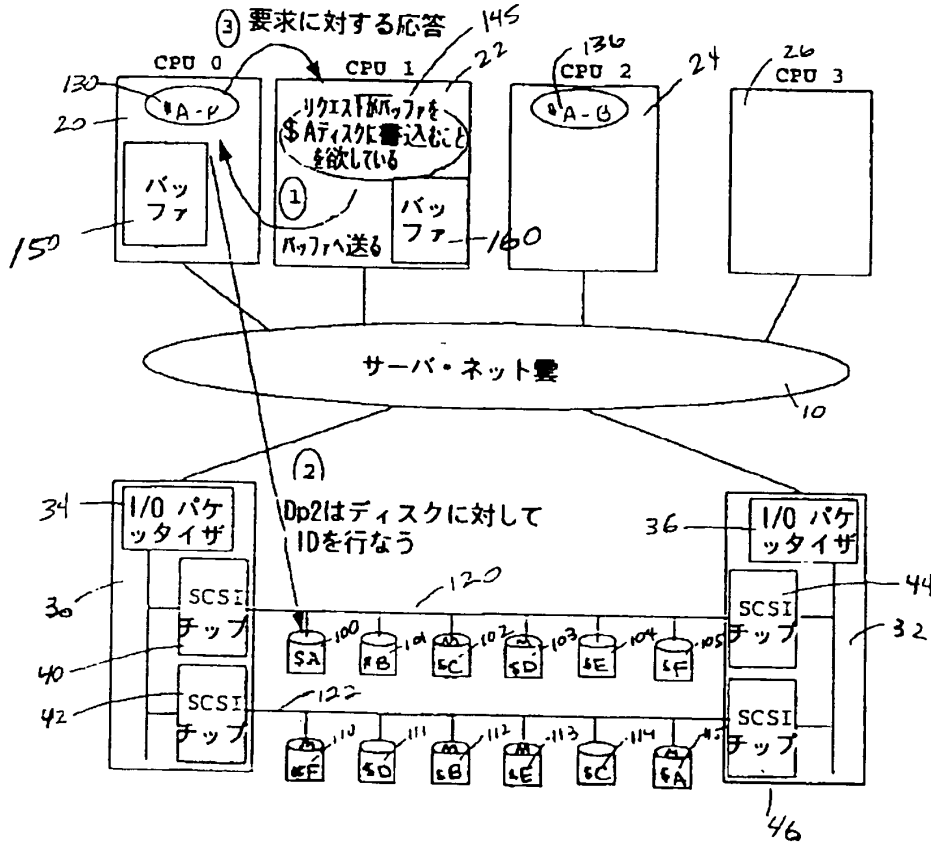
【図1】

[Figure 1]



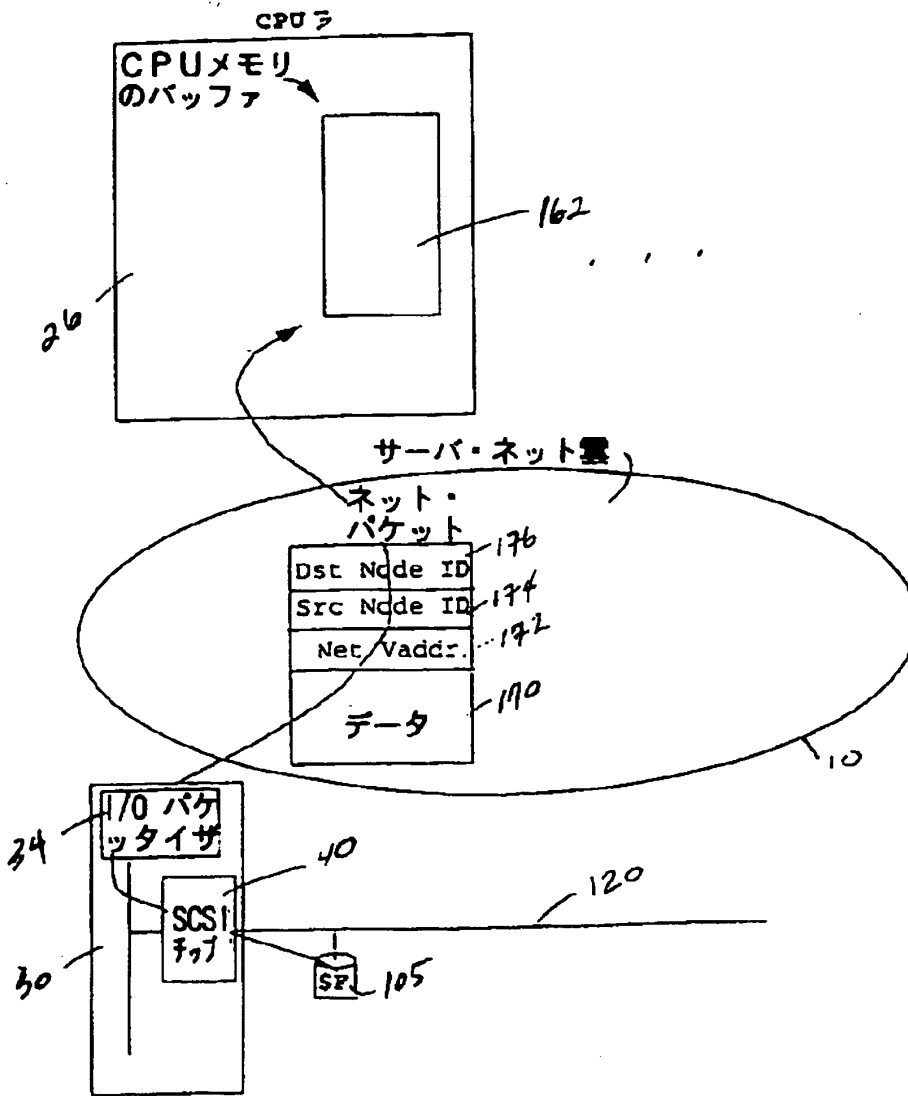
【図2】

[Figure 2]



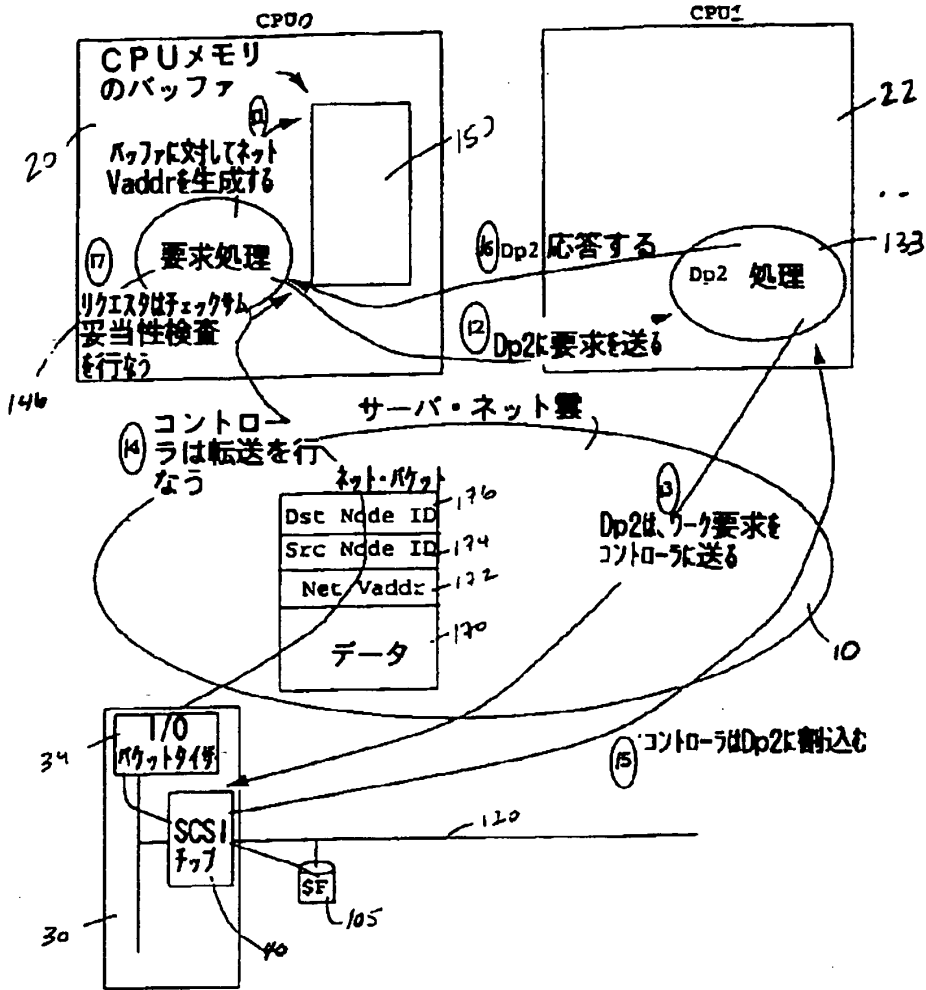
【図3】

[Figure 3]



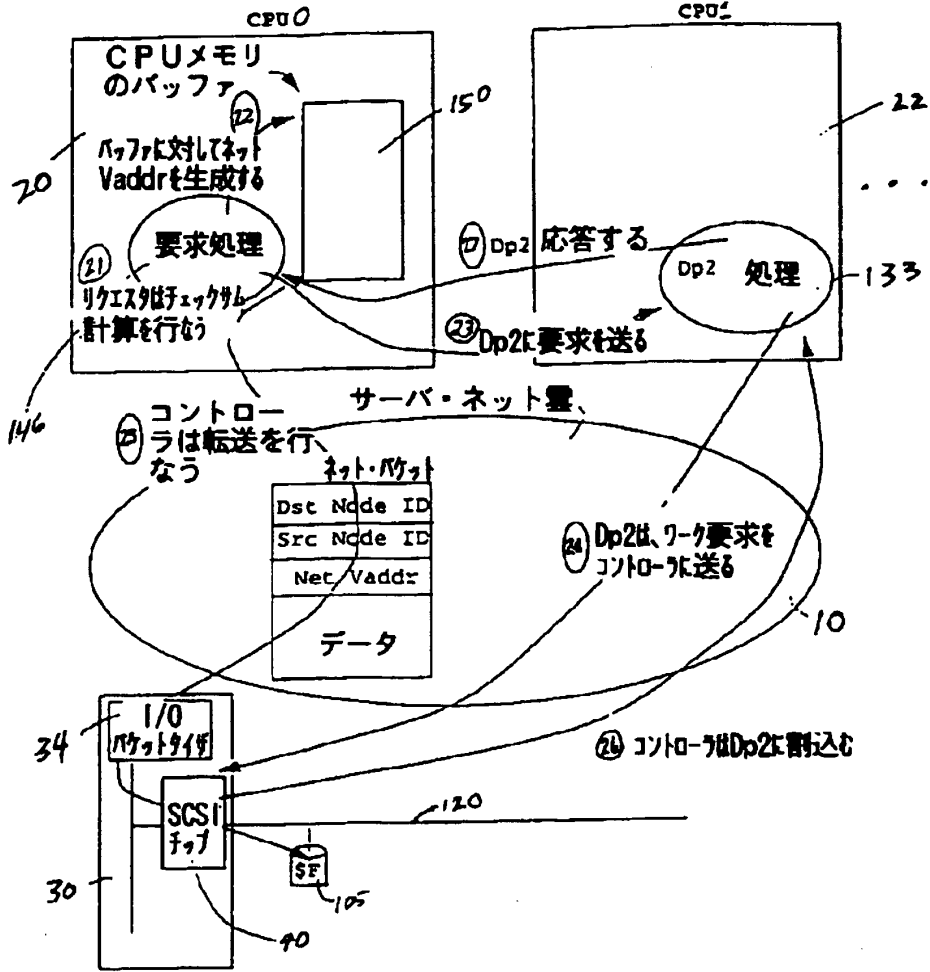
【図4】

[Figure 4]



【図5】

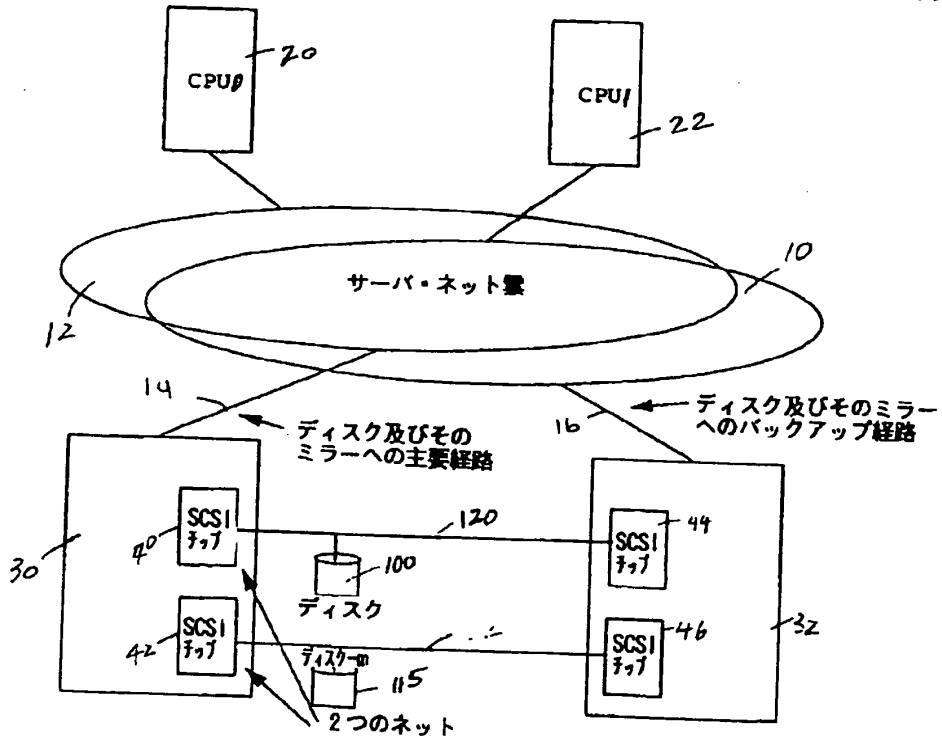
[Figure 5]



【図6】

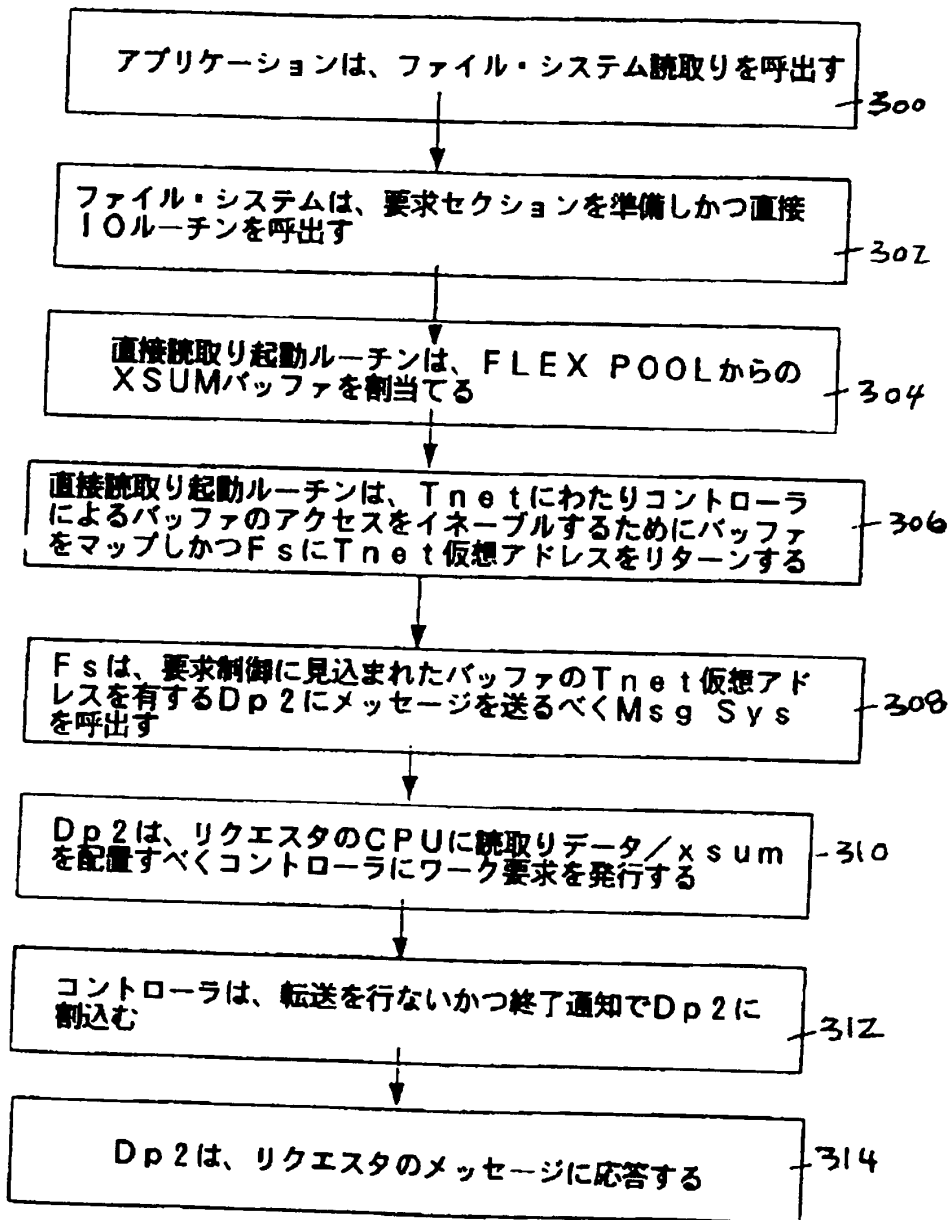
[Figure 6]





【図8】

[Figure 8]



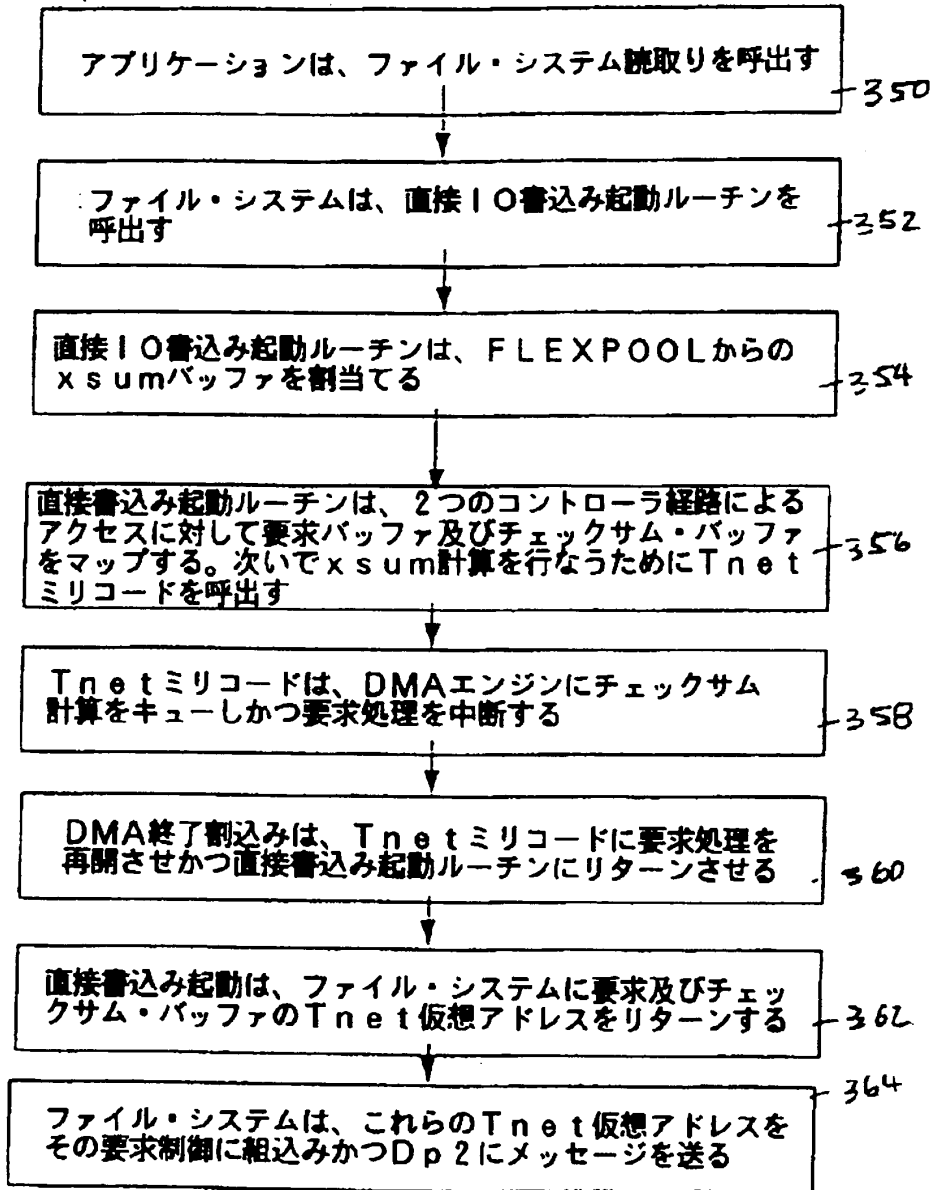
【図9】

[Figure 9]



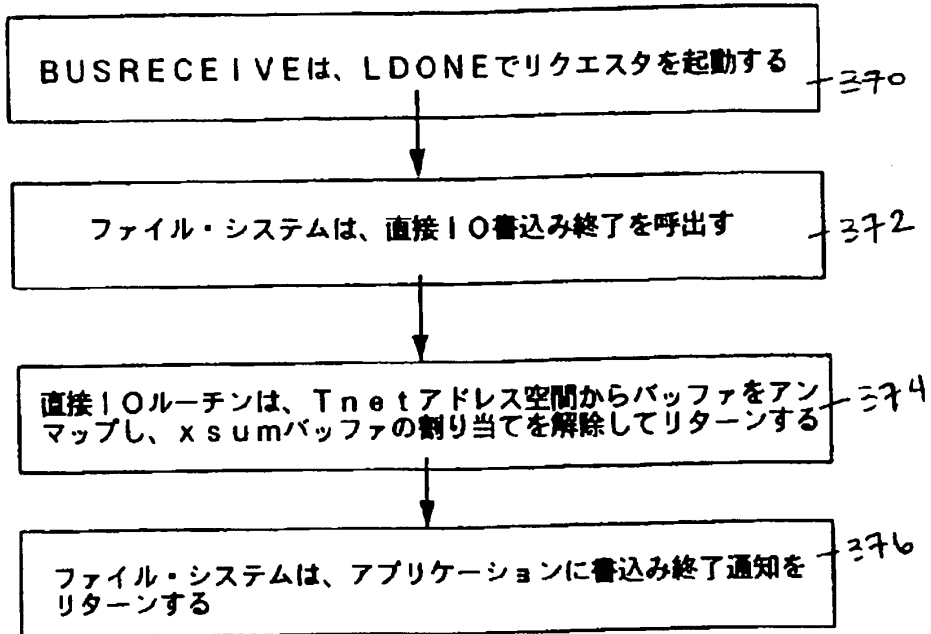
【図10】

[Figure 10]



【図11】

[Figure 11]



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-185594

(43) 公開日 平成9年(1997)7月15日

(51) IntCl. <sup>6</sup> G 0 6 F 1 5/1 6 3	識別記号 庁内整理番号	F I G 0 6 F 1 5/1 6	技術表示箇所 3 2 0 R 3 2 0 V
---	----------------	------------------------	------------------------------

審査請求 未請求 請求項の数6 OL (全25頁)

(21) 出願番号 特願平8-301770

(22) 出願日 平成8年(1996)11月13日

(31) 優先権主張番号 08/556618

(32) 優先日 1995年11月13日

(33) 優先権主張国 米国 (US)

(71) 出願人 391058071  
 タンデム コンピューターズ インコーポ  
 レイテッド  
 TANDEM COMPUTERS IN  
 CORPORATED  
 アメリカ合衆国 カリフォルニア州  
 95014 クーパーティノ ノース タンタ  
 ウ アベニュー 10435

(72) 発明者 トッド ダブリュー スプレングル  
 アメリカ合衆国 カリフォルニア州  
 94086 サニーヴェイル ヴァスケズ ア  
 ベニュー 387

(74) 代理人 弁理士 中村 総 (外6名)

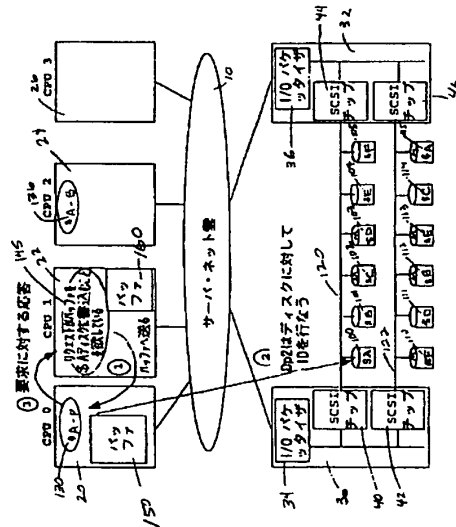
最終頁に続く

(54) 【発明の名称】 直接バルク・データ転送

(57) 【要約】 (修正有)

【課題】 CPUsと記憶ディスクとの間のデータの転送中の類似するデータ・コピーを除去する。

【解決手段】 記憶処理130は、記憶装置100~105及び110~115を有する30へのアクセスを制御する。ソフトウェア・ルーチンは、要求CPU22による記憶装置30への直接アクセスを供給するために用いられる。要求CPU22のバッファ160に対する仮想メモリ・アドレスは、要求CPU22で生成される。記憶装置アクセス要求と共に仮想メモリ・アドレスは、記憶処理130を含んでいるCPU20に送られる。ワーク要求は、記憶処理130から記憶装置30へ送られる仮想メモリ・アドレスを含む。次いで、データは、要求CPU22と記憶装置30との間で直接転送される。記憶装置30は、次いでワーク要求に応答する。



【特許請求の範囲】

【請求項1】 データを転送するデータ処理システムであって、

要求CPUを含んでいる複数の中央処理装置(CPU s) ;前記CPU sの一つが記憶装置へのアクセスを制御する少なくとも一つの記憶装置 ;前記複数のCPU sと前記記憶装置とを相互接続するネットワーク ;前記要求CPUによる前記記憶装置の直接アクセスを供給し、前記要求CPUのバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に前記仮想メモリ・アドレスを、前記記憶装置へのアクセスを制御している前記CPU sの前記一つに供給する前記要求CPUにおける手段、前記ワーク要求に反応しておりかつ前記ネットワークを通して前記CPU sの前記一つと直接インターフェイスしている前記記憶装置に前記仮想メモリ・アドレスを含んでいるワーク要求を送る前記CPU sの前記一つにおける手段、を含んでいるアクセス手段 ;を備え、

前記データは、前記要求CPUと前記記憶装置との間で直接転送されることを特徴とするデータ処理システム。

【請求項2】 前記直接アクセスは、前記要求CPUに前記記憶装置から前記仮想メモリ・アドレスにおける前記バッファ・メモリの中にデータを読取らせることを特徴とする請求項1に記載のデータ処理システム。

【請求項3】 前記データは、それぞれが、宛先ノード識別、ソース・ノード識別、前記仮想メモリ・アドレス、及び複数のデータ・ワードを含んでいるデータ・パケットで送信されることを特徴とする請求項1に記載のデータ処理システム。

【請求項4】 前記記憶装置は、転送のためのデータを含んでいる前記データ・パケットのそれぞれが前記要求CPUに送信された後で前記CPU sの前記一つにアドレスをする手段を含むことを特徴とする請求項3に記載のデータ処理システム。

【請求項5】 前記直接アクセスは、前記要求CPUに、前記データの転送のために前記仮想メモリ・アドレスにおける前記バッファをアクセスしている前記記憶装置の中に書込ませることを特徴とする請求項1に記載のデータ処理システム。

【請求項6】 前記アクセス手段は、前記要求CPUの前記バッファに対する前記仮想メモリ・アドレスを生成しかつ前記記憶装置アクセス要求と共に前記仮想メモリ・アドレスを前記CPU sの前記一つに供給する少なくとも一つのソフトウェア・ルーチンを含むことを特徴とする請求項1に記載のデータ処理システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、一般にデータ処理システムにおける直接データ転送に関し、より特定的にはプロセッサ及び入力/出力通信に対する接続性を供給する

ネットワークを介してシステムで行われる直接バルク入力/出力転送に関する。

【0002】

【従来の技術】サード・パーティ転送システムは、データ転送に対する種々のスキームを供給する。例えば、高性能記憶システム(High-Performance Storage System (HPSS))は、高集合体I/O処理能力(スループット)を達成するためにネットワークにわたり同時入力/出力(I/O)オペレーションを統合(調整)することができる高度な、分散型階層記憶管理システムである。HPSSは、国立記憶研究所(National Storage Laboratory)で開発中の次世代ソフトウェア・システムである。HPSSnについての更なる情報は、次に示すインターネット上のワールド・ワイド・ウェブ・ページを通して取得することができる :

[http://www.ccs.ornl.gov/HPSS/HPSS\\_\\_overview.html](http://www.ccs.ornl.gov/HPSS/HPSS__overview.html)

[http://www.llnl.gov/liv\\_\\_comp/ns1/hpss/hpss.html](http://www.llnl.gov/liv__comp/ns1/hpss/hpss.html)

この情報は、HPSSのソース・デバイス(source device)からシンク・デバイス(sink device)までデータを転送するために用いられるムーバ(Mover)を開示する。このムーバは、また、一組のデバイス制御オペレーションを実行する。

【0003】ファイバ・チャネル(FC)は、また、サード・パーティ転送を供給する。FCは、ワークステーション、メインフレーム、スーパーコンピュータ、デスクトップ・コンピュータ、記憶装置、ディスプレイ及び他の周辺装置間でデータの転送を許容する。FCについての更なる情報は、次に示すインターネット上のワールド・ワイド・ウェブ・ページを通して取得することができる :

<http://www.amdahl.com/ext/CARP/FCA/FCA.html>

そして、IEEE記憶システム標準研究グループは、オープン記憶システム相互接続(Open Storage Systems Interconnection(OSSI)) に対するドラフト 標準モデル(draft Reference Model) を生成した。次に示すのは、ドラフトOSSIドキュメントのバージョン5からのものである :

“2.3.3 制御及びデータ・フローの分離

“OSSIモデルは、クライアント、データ・ソース、及びデータ・シンクの間で発生しているデータ・フローから制御フローを区別する。制御フローは、クライアントとデータ・ソースまたはデータ・シンクとの間で要求、応答、及び非同期通知をキャリアする。データ・ソースとデータ・シンクとの間の制御フローは、データのフローを管理するためにソース-シンク・プロトコル情報をキャリアする。データ・フローは、ソースからシンクまでのみパスする。制御及びデータ・フローを論理的に分離することによって、OSSIモデルは、個別のインプリメンテーションを通る各フローを最適化することの可能性を供給する。”

“2.3.4 サード・パーティ転送

“OSSIモデルは、サード・パーティの制御下で、データ

を独立のソース及びシンク間で直接フローさせ、エージェントまたはクライアントを起動しかつ制御する。各エンティティは、データ・フロー制御、エラー報告、または転送を開始しかつ終了することのようなオペレーションを個別に実行する。”

OSSIについての更なる情報は、次のワールド・ワイド・ウェブ・ページを介して取得することができる：

<http://www.arl.mil/IEEE/ssswq.html>

全ての目的のためにここに参照文献として採り入れた、1995年6月7日に開示された米国特許出願第 08/48 6,217 号 (添付資料Aとして添付される)は、プロセッサ及び入力/出力通信間の接続性または結合性を供給する高信頼システム・エリア・ネットワークにおける多重処理システムに対する必要性に因ずる。この特許出願 (以下、添付特許出願と呼ぶ)は、フェール・ファースト、フェール・ファンクショナル、フェール・トレラント・マイクロプロセッサ・システムを供給する。添付特許出願に開示されたアーキテクチャは、あらゆる中央処理装置 (CPU) を入力/出力 (I/O) コントローラ (サーバ・ネット・アダプタとも呼ばれる) と通信させるサーバ・ネットワーク雲 (server net cloud) を含む。従って、I/O コントローラは、CPU によってアドレス指定されることができ、かつ CPU は、I/O コントローラによってアドレス指定されることができる。このサーバ・ネット雲は、添付特許出願ではサーバ・ネットと呼ばれている。本願と共に添付特許出願は、タンデム・コンピュータ・インコーポレーティッドに譲渡されている。

[0004] 広く言えば、添付特許出願に開示された発明は、多重サブ処理システムを備えた処理システムを含む。各サブ処理システムは、主処理構成要素として、CPU を有する。この CPU は、同時に命令ストリームからの各命令を実行するためにロック・ステップ同期 (lock-step synchronized) 方式で動作する一対のプロセッサを含む。サブ処理システムのそれぞれは、より大きな処理システムの種々のコンポーネント間に冗長通信経路を供給する I/O システム・エリア・ネットワークを更に含む。これらの種々のコンポーネントは、CPU 及びアサートされた周辺デバイス (例えば、マス記憶装置、プリンタ、等) を含む。これらの冗長通信経路は、また、より大きな総括処理システムを形成するサブプロセッサ間に存在することができる。処理システムのコンポーネント間 (例えば、第1のCPUと第2のCPUの間、またはCPUと周辺デバイスとの間) の通信は、パケットに含まれるメッセージを形成しかつ送信することによってインプリメントされる。好ましい実施例では、各パケットは、64バイトのデータを含む。これらのパケットは、システム・エリア・ネットワーク構造 (サーバ・ネットと呼ぶ) により送信またはソース・コンポーネント (例えば、CPU) から宛先構成要素 (例えば、周辺

デバイス) へ送られる。

[0005] このシステム・エリア・ネットワーク構造は、複数の相互接続リンクによって相互接続される多数のルータ構成要素を含む。パケットのソースからパケットの宛先へパケットを送る、添付特許出願に開示されたルータは、それら自身パケットを発生しない。ルータは、一つのリンク上の入力パケットを取り入れてかつそれをその宛先に対して適切なリンク上に送り出すことによってパケット・スイッチとして動作する。ルータ構成要素は、処理システムの送信コンポーネントから宛先コンポーネントまでの適切なまたは利用可能な通信経路を選択する責任がある。通信経路は、メッセージ・パケットに含まれる情報に基づく。それゆえに、ルータ構成要素のルーティング能力 (ケイバビリティ) は、周辺デバイスへの通信経路を有する CPU の I/O システムを供給する。

[0006] 添付特許出願に開示されたアーキテクチャは、各ディスクを管理するためにディスク処理ペアを用い、ディスク処理ペアの半分は、1次ディスク処理でありかつ他の半分は、バックアップ・ディスク処理である。更に、SCSIチェイン上のディスクを制御しているディスク処理は、二つのCPUに制限されず、かつディスク処理は、複数のCPUの中で走るように構成することができる。添付特許出願のサーバ・ネット雲が用いられるときには、CPU及びサーバ・ネット・アダプタの両方は、CPUメモリに対する読取り及び書込みサーバ・ネット雲トランザクションを発生することができる。図1は、記憶装置のアーキテクチャの一例を示す。この構成は、添付特許出願に開示されたサーバ・ネット雲10を含む。ディスク/記憶コントローラ30及び32と共にCPU20、22、24及び26は、サーバ・ネット雲10に接続される。サーバ・ネット・アダプタ30及び32は、SCSIチップ40、42、44及び46と共にI/Oパケットタイザ(packetizers) 34及び36を含む。I/Oパケットタイザは、ネットワーク・プロトコルからのデータ・パケットをバス・プロトコルに変換する。この構成は、また、(合計12の記憶ディスク100~105及び110~115に対して) 二つのSCSIチェイン120及び122をハング・オフしている(hanging off) 6つのSCSIディスク・ペア100~105及び110~115を示す。ディスクは、1次ディスク100 (\$A)、101 (\$B)、104 (\$E)、105 (\$F)、111 (\$D) 及び114 (\$C) 及びミラー・ディスク102 (\$C)、103 (\$D)、110 (\$F)、112 (\$B)、113 (\$E) 及び115 (\$A) として構成される。4つのCPU (CPU0 20、CPU1 22、CPU2 24及びCPU3 26) は、6つのディスク・ペア100~105及び110~115を制御するディスク処理を収容する。1次ディスク処理 (\$A-P、\$B-P、\$C-



P、\$D-P、\$E-P及び\$F-P)及びバックアップ・ディスク処理(\$A-A、\$B-B、\$C-B、\$D-B、\$E-B及び\$F-B)は、4つのCPU s 20、22、24及び26の中に散乱される。例えば、記憶ディスク100(\$A)に対する1次ディスク処理130(\$A-P)は、CPU0 20に配置されうるし、かつ記憶ディスク102(\$C ミラー)に対するバックアップ・ディスク処理133(\$C-P)は、CPU1 22に配置されうる。ディスク処理130~141は、二つのCPU s以上に(図1に示すように)配置することができる。別の構成では、8つの記憶ディスク・ペアは、(合計16の記憶ディスクに対して)二つのSCSIチェーンをハング・オフすることができ、かつエキストラ(余分な)SCSIチップは、外部記憶デバイスに対する各サーバ・ネット・アダプタに配置することができる。

【0007】添付特許出願に示したように、ディスク100に書き込むためのCPU1 22に配置される要求処理145に対して、要求処理145は、書き込みデータ・メッセージをCPU0 20に配置されたディスク処理130にまず送る(ディスク処理130は、ディスク100を制御する)。次いで、ディスク処理130は、データにわたりチェックサムを計算する。チェックサムは、データのブロックに対するデータ安全性が転送されることを確実にする。次いで データのブロックは、サーバ・ネット雲10を通してCPU0 20からディスク100に転送される。この転送が終了したときに、ディスク処理130は、要求処理145に応答する。

【0008】

【発明が解決しようとする課題】図2は、ディスク転送の一例を示す。図2は、図1に含まれる同じ構成要素のほとんどを有する。これらの構成要素に加えて、CPU0 20に配置されるバッファ150及びCPU1 22に配置されるバッファ160が示されている。この例では、要求処理及びディスク処理は、異なるCPU sに配置される。これらの処理は、また、同じCPUに配置されることもできる。段階1では、要求処理は、バッファ160から\$Aディスク100にデータを書き込むことを欲する。従って、CPU0 20に配置された、ディスク処理130が\$Aディスク100に対するディスク処理なので、バッファ160に配置されたデータは、CPU0 20に配置されたバッファ150に送られる。段階2では、ディスク処理130は、バッファ150に配置されたデータをディスク100に書き込む。段階3では、ディスク100へのデータの転送が終了したので、ディスク処理130は、要求処理145に応答する。

【0009】この構成では、データは、要求処理を有するCPUから関連ディスク処理を有する中間CPUにコピーされ、そしてその中間CPUからディスクにコピーされる。CPU sと記憶ディスクとの間のデータの転送

の間中、類似するデータ・コピーを除去することが望ましい。本発明の目的は、上記従来技術の問題に鑑み、CPU sと記憶ディスクとの間のデータの転送中に類似するデータ・コピーを除去することができるシステムを提供する。

【0010】

【課題を解決するための手段】本発明の上記目的は、データを転送するデータ処理システムであって、要求CPUを含んでいる複数の中央処理装置(CPU s)；CPU sの一つが記憶装置へのアクセスを制御する少なくとも一つの記憶装置；複数のCPU sと記憶装置とを相互接続するネットワーク；要求CPUによる記憶装置の直接アクセスを供給し、要求CPUのバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に仮想メモリ・アドレスを、記憶装置へのアクセスを制御しているCPU sの一つに供給する要求CPUにおける手段、ワーク要求にตอบสนองしておりかつネットワークを通してCPU sの一つと直接インターフェイスしている記憶装置に仮想メモリ・アドレスを含んでいるワーク要求を送るCPU sの一つにおける手段、を含んでいるアクセス手段；を備え、データは、要求CPUと記憶装置との間で直接転送されるデータ処理システムによって達成される。

【0011】本発明のデータ処理システムでは、直接アクセスは、要求CPUに記憶装置から仮想メモリ・アドレスにおけるバッファ・メモリの中にデータを読み取らせるようにしてもよい。本発明のデータ処理システムでは、データは、それぞれが、宛先ノード識別、ソース・ノード識別、仮想メモリ・アドレス、及び複数のデータ・ワードを含んでいるデータ・パケットで送信されるようにしてもよい。本発明のデータ処理システムでは、記憶装置は、転送のためのデータを含んでいるデータ・パケットのそれぞれが要求CPUに送信された後でCPU sの一つにアドバイスをする手段を含むようにしてもよい。本発明のデータ処理システムでは、直接アクセスは、要求CPUに、データの転送のために仮想メモリ・アドレスにおけるバッファをアクセスしている記憶装置の中に書き込ませるようにしてもよい。

【0012】本発明のデータ処理システムでは、アクセス手段は、要求CPUのバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に仮想メモリ・アドレスをCPU sの一つに供給する少なくとも一つのソフトウェア・ルーチンを含むようにしてもよい。

【0013】

【作用】本発明は、データを転送するためのデータ処理システムを供給する。このシステムは、ネットワークによって相互接続された中央処理装置(CPU s)及び記憶装置を含む。CPU sは、要求処理及び記憶処理(ディスク処理とも呼ばれる)を含む。記憶処理は、記憶装

置へのアクセスを制御する。ソフトウェア・ルーチンは、要求CPU（要求処理を含んでいるCPU）による記憶装置への直接アクセスを供給するために用いられる。要求CPUのバッファに対するサーバ・ネット仮想メモリ・アドレスは、要求CPUに生成される。記憶装置アクセス要求と共にサーバ・ネット仮想メモリ・アドレスは、記憶処理を含んでいるCPUに送られる。サーバ・ネット仮想メモリ・アドレスを含んでいるワーク要求は、記憶処理から記憶装置に送られる。次いでデータは、要求CPUと記憶装置との間に直接転送される。記憶装置は、次いで、ワーク要求にตอบสนองする。

【0014】本発明の更なる態様及び特徴は、添付した図面に関して取入れられるべき、本発明の以下の詳細の説明を読むことにより当業者に明らかになるであろう。

【0015】

【実施例】本発明は、プロセッサ及びI/O通信に対する接続性を供給するネットワークを介して高信頼システムにおける直接バルク・データ転送を供給する。この直接データ転送は、要求処理を走らせているCPUとディスク処理を走らせているCPUとの間のデータの複写を除去すると同時にディスク処理へ構成を維持する。それゆえに、データは、要求処理のバッファと記憶装置との間で直接複写される。結果として、(1)データがディスク処理のバッファに転送されないでネットワーク帯域幅は、セーブされ、(2)ディスク処理がそのバッファの中にデータを受け取らなくともよいのでコンテキスト・スイッチ時間がセーブされ、(3)バッファ複写が回避されるので、入力/出力(I/O)ラテンシーは、低減され、かつ(4)メッセージ・システムが要求処理とディスク処理との間でデータを送らないのでワークは、メッセージ・システムから“負担が軽減される”。このメッセージ・システムは、プロセッサ間通信に用いられる。

【0016】図3は、コントローラが“プッシング”(pushing)しているときに発生するデータの転送を示している。例えば、データがディスク105からCPUメモリへ転送されるときに、サーバ・ネット・アダプタ30は、CPUメモリにデータ170を“プッシング”する。これは、また、“読取り”オペレーションとも呼ばれる。この構成では、要求処理及びデータ処理の両方は、CPU326に配置されている。このCPU326は、サーバ・ネット・アダプタ30に、データ転送に関連付けられたパラメータ（例えば、遠隔ノード識別）と共に、データ170を受取る、バッファ162に対するサーバ・ネット仮想アドレス172を送る。サーバ・ネット仮想アドレス172は、バッファ162の物理アドレスを取得するために用いる情報を含む。このサーバ・ネット仮想アドレス172は、後ほど物理アドレスの中に交換して戻される。好ましい実施例では、サーバ・ネット仮想アドレス172は、頁番号と、関連物理

アドレスを決定するために一緒に用いられることが必要であるオフセットとの両方を含む。従って、二つ以上のサーバ・ネット仮想アドレス172を一つのバッファに対して設けることができる。サーバ・ネット仮想アドレスは、全ての目的に対して参考文献として採り入れられた添付特許出願により詳細に開示されている。バッファ162は、要求処理に属する。転送に関連付けられたパラメータは、“ワーク要求”(work request)を介してCPU326からサーバ・ネット・アダプタ30に送られる。この情報で、サーバ・ネット・アダプタ30は、I/Oデバイス（例えば、\$Fディスク105）から要求処理のメモリに直接データ170を転送することができる。同様な構成で、CPUは、I/Oデバイス（ディスク）に“書込む”ことができる。これは、サーバ・ネット・アダプタがCPUメモリからデータを“プッシング”(pulling)している呼ばれる。この構成では、CPU326は、I/O空間に関連付けられたパラメータと共に、データを含んでいるCPUバッファ162に対するサーバ・ネット仮想アドレス172をサーバ・ネット・アダプタ30に送る。再び、これらのパラメータは、“ワーク要求”に含まれる。この情報で、サーバ・ネット・アダプタ30は、CPUメモリからディスク・デバイス105にデータを転送することができる。従って、データは、非ディスク処理を有するCPUと、関連ディスク処理を走らせているCPUを通してデータを送らないディスクとの間で転送される。

【0017】サーバ・ネット雲を通して送られたパケット（サーバ・ネット・パケットと呼ばれる）は、サーバ・ネット・パケットに対するソース・ノード174及び宛先ノード176を識別するヘッダを含む。この識別は、ノード識別(ID)の形である。好ましい実施例では、CPU及び他の周辺デバイスだけがノードIDを有する。各サーバ・ネット・パケット・ヘッダは、また、パケットの型を示すトランザクション型フィールドを含む。パケット型が読取られるかまたは書込まれるときに、パケットは、宛先ノードから肯定応答(acknowledgement)を導き出す。読取りパケットに対して、この肯定応答は、読取りを実行しているノードにリターンされるべきデータを含む。書込みパケットに対して、この肯定応答は、書込みオペレーションに対する成功または失敗状態をリターンする。

【0018】サーバ・ネット読取り/書込みパケットで指定されたサーバ・ネット・アドレスは、宛先ノードにおける物理アドレスではなく、その代わり、許可チェックされそして、許可チェックがパスしたならば、物理アドレスに変換される、アドレスである。このサーバ・ネット・アドレスは、上記で参照したサーバ・ネット仮想アドレスである。許可検査は、例えば、ソース・ノード識別妥当性検査、変換型検査（即ち、読取り/書込み許可）及びバウンド(bounds)検査の形である。アドレス妥

当性検査及び変換 (AVT) 表は、サーバ・ネット仮想アドレスに適用された変換及び許可検査に用いられる。ソース・ノードIDを確認 (検査) するために、アクセスしたAVTエントリのソースIDフィールドは、用いられるAVTエントリに対応するソースを指定する。このソースIDフィールドは、不整合(mismatch)がアクセスを否定するAVTエラー割込みを結果として生ずるように、要求しているメッセージ・パケットに含まれるソースIDと比較される。トランザクション型検査は、読取りまたは書込みのためのアクセスが許可されることが

できるかどうかを決定するための検査を含む。バウンド検査に対して、AVTエントリの下部バウンド・フィールド及び上部バウンド・フィールドは、アクセスが許可されるかどうかを決定するためにオフセット値と比較される。この型の許可検査は、添付特許出願 (例えば、その頁55~61、図13A~13C) に詳細に記載されている。

【0019】データの転送を達成するために用いられるハードウェア直接メモリ・アクセス (DMA) エンジン

は、また、ブロック転送エンジンとも呼ばれる。好ましい実施例では、ブロック転送エンジンは、非同期でメモリ・バッファ上でチェックサムを計算することができ、かつ少なくとも一つのDMAエンジンが各CPUに配置される。上記したように、チェックサムは、転送されるデータのブロックに対してデータの貫性を確保にする。上記したように、CPU20、22、24または26は、サーバ・ネット雲にわたりデータのバケットを転送するためにブロック転送エンジンにバッファを委ねることができる。更に、CPU20、22、24または26は、チェックサムを計算しかつ別のバッファにチェックサムを配置するためにブロック転送エンジンにバッファを委ねることができる。そして、好ましい実施例では、SCSIチップ40に配置されたDMAエンジンは、(ディスク“読取り”または“書込み”に対して) 各データ転送に対して“プッシング” (pushing) または“プリング” (pulling) を実行する。

【0020】上記したように、本発明の直接データ転送が行われるときに、データは、要求発生CPUとディスク処理を含んでいるCPUとの間で転送されない。代わりに、サーバ・ネット仮想アドレスが生成されかつ用いられる。このサーバ・ネット仮想アドレスは、直接データ転送を実行するためにサーバ・ネット・アダプタによって用いられる。再び、この処理において二つの型のデータ転送が存在する。第1は、サーバ・ネット仮想アドレスを生成すること及び用いることを含むディスク書込みオペレーションである。サーバ・ネット・アダプタは、ディスク書込みに対して必要なオペレーションを行うためにこのサーバ・ネット仮想アドレスを用いる。それゆえに、(要求処理バッファに対する) サーバ・ネット仮想アドレスに対するAVTエントリは、要求処理バ

ッファをアクセスすることを必要とするサーバ・ネット・アダプタに対するサーバ・ネット・ノードIDを指定する。

【0021】ディスク書込みオペレーションに対して、要求処理を走らせている発生CPUは、ディスク処理への要求を発行する前にデータのチェックサムをまず計算する。要求処理がそのバッファに対してサーバ・ネット仮想アドレスを生成した後、要求処理は、そのサーバ・ネット仮想アドレスをディスク処理にパスする。ディスク処理は、ソースID妥当性検査が失敗しうるので、要求処理バッファをアクセスするためにこのサーバ・ネット仮想アドレスを用いることができない。サーバ・ネット仮想アドレスを受け取った後、ディスク処理は、

(1) 要求を発生しているCPU (要求処理を含んでいるCPU) に対するサーバ・ネット・ノードIDへの処理、及び(2) 要求処理に関連付けられたバッファに対するサーバ・ネット仮想アドレス、を指定しているサーバ・ネット・アダプタへ“ワーク要求”を送る。サーバ・ネット・アダプタは、それが書込みオペレーションであるので、サーバ・ネット仮想アドレスを用いてデータを次いで“プル” (pulls) する。このデータの転送の終りで、(一般に割込みの形の) 通知が、ディスク処理を含んでいるCPUに送信して戻される。次いで、ディスク処理は、データ転送の終了をそれに通知するために要求処理に応答する。AVT表及びサーバ・ネット仮想アドレスに関する更なる情報は、添付特許出願に供給されている。

【0022】第2の型のデータ転送は、ディスク読取りに対してである。図4は、ディスク読取りに対する直接データ転送の一例を示している。段階11では、バッファ150に対するサーバ・ネット仮想アドレスが生成される。バッファ150は、要求処理146に関連付けられる。段階12では、要求処理146は、要求をディスク(Dp2) 処理133へ送る。段階13では、ディスク処理133は、ワーク要求をサーバ・ネット・アダプタ30へワーク要求を送る。段階14では、サーバ・ネット・アダプタ30は、\$Fディスク105からバッファ150へのデータの転送を実行する。段階14は、要求データの全てがバッファ150へ送られるまで行われる。データ転送が終了したときには、サーバ・ネット・アダプタ30は、段階15でディスク処理133に割り込む。段階16では、ディスク処理133は、データ転送が終了したことを要求処理146に知らせる。段階17では、要求処理を走らせている発生CPUは、バッファのチェックサムを計算しそしてデータを受け入れる前にそれを確認する。

【0023】ディスク書込みに対する直接データ転送は、(1) 要求がディスク処理に送られる前にチェックサムが要求処理によって計算され、かつ(2) データの転送の方向がCPUメモリからディスクへであることを

除き、ディスク読取りに非常に類似している。図5は、ディスク書込みに対する直接データ転送の一例を示している。段階21では、要求処理146は、チェックサム計算を実行する。段階22では、要求プロセッサ146は、バッファ150に対しかつチェックサム・バッファに対してサーバ・ネット仮想アドレスを生成する。段階23では、要求処理146は、その要求をディスク処理133へ送る。段階24では、ディスク処理133は、ワーク要求をサーバ・ネット・アダプタ30へ送る。段階25では、サーバ・ネット・アダプタ30は、バッファ150からのデータ及びチェックサム・バッファからのチェックサムを\$Fディスク105へ転送する。要求したデータの全てがバッファ150からディスク105へ転送された後、サーバ・ネット・アダプタ30は、段階26でディスク処理133に割込む。段階27では、ディスク処理133は、データ転送が終了したことを要求処理146に知らせる。

【0024】直接データ転送に対するソフトウェアは、ディスク・デバイスからの(または、に)要求処理を走らせているCPUに(または、から)データを直接送らせる。好ましい実施例において直接データ転送を実行するために、デバイス・ハンドルが用いられる。このデバイス・ハンドルは、遠隔サーバ・ネット・ノードIDがデータ転送に対するAVTエントリに供給する。このノードIDによって識別されたサーバ・ネット・アダプタのみがデータ転送に対するAVT表におけるエントリをアクセスすることができる。AVT表は、それが正しいノードIDを有するサーバ・ネット・アダプタにアクセス可能であるように、要求CPUのバッファをサーバ・ネット仮想アドレス空間にマップする。起動されたときに、デバイス・ハンドルは、ノードIDを含むデータ転送パラメータを指し示す。このデバイス・ハンドルを取得するために、“TSER\_DEVINSTALL”とレッテルを貼られたサーバ・ネット・ノード・ルーチンが好ましい実施例で用いられる。従って、遠隔サーバ・ネット・ノードに関連付けられたパラメータは、このルーチンが呼出されるときにコードに指定される。

【0025】好ましい実施例では、システムは、データ転送に対する要求が認識されるときに直接データ転送を起動しかつTSER\_DEVINSTALLを実行する。直接データ転送処理が終了したときに、デバイス・ハンドルは、例えば、TSER\_DEVREMOVE ルーチンを呼出すことによってリターンされる。第2の実施例では、デバイス・ハンドル変換キャッシュに対する汎用(グローバル)デバイス・ハンドルまたはスタック/モジュール/スロットが各CPUに配置される。この実施例では、TSER\_DEVINSTALL及びTSER\_DEVREMOVE ルーチンは、呼び出されない。ユーザは、多くの方法で直接データ転送を用いることの要望を示すことができる。例えば、SETMODE 141は、ディスク・ファイル転送に対する大きなデータ転送モード

をイネーブ爾できる。このSETMODE は、ユーザに、ディスク処理キャッシュをバイパスするディスク・ファイルへの大きな、非構成アクセスを彼らが要望することを指定されることができる。別の例では、BULKREAD及びBULKWRITEは、バックアップ、復元、ダンプ、等が要求されるときに、内部ツールとして用いることができる。BULKREAD及びBULKWRITE は、また、ディスク処理キャッシュもバイパスする。それゆえに、BULKREADまたはBULKWRITE 或いはSETMODE 141がファイル転送に対してイネーブ爾されたならば、直接データ転送は、転送タスクを実行するために用いられる。

【0026】好ましい実施例では、SETMODE 141が示されたときに、メッセージは、そのキャッシュをフラッシュするためにディスク処理に送られる。このSETMODE 要求への応答の一部として、ディスク処理は、それが直接データ転送を支持することができるファイル・システムに対して指示(indication)をリターンする。ディスク処理は、また、二つのサーバ・ネットIDをファイル・システムにリターンする。これらのサーバ・ネットIDは、関連ディスク及びそのミラー・ディスクへの二つの主要(1次)経路を構成する。ディスク・デバイスへの書込みを行うときに両方のサーバ・ネットIDが用いられ、かつ一つのサーバ・ネットIDのみが読取りオペレーションに対して用いられる。ファイル・システムがディスク処理から応答を受け取るときに、それは、応答を解読しかつサーバ・ネット・コードを二つのサーバ・ネットIDに設置するために直接データ転送ルーチンを呼出す。

【0027】同様に、BULKREADまたはBULKWRITE が指示されたときに、ファイル・システムは、通常のバルク・データ転送要求をディスク処理に送る。可能な場合には、ディスク処理は、それが直接データ転送を支持することができることを示しているファイル・システムに回答する。この時には、ディスク処理は、また、関連ディスク及びそのミラーへの二つの主要経路を構成する二つのサーバ・ネットIDをリターンする。次いで、ファイル・システムは、後続のバルク・データ転送呼出しに対して直接データ転送を用い、かつまたサーバ・ネット・コードを二つのサーバ・ネットIDに設置するために直接データ転送ルーチンを呼出す。(二つサーバ・ネットIDを設置する)直接データ転送セッション確立の一部として、直接データ転送ルーチンは、また、FLEXPOOLからの転送情報ブロック(TIB)に対する空間も獲得する。TIBは、直接データ転送ルーチンとサーバ・ネット・コードとの間のインターフェイスに用いられる。TIBは、また、チェックサム計算を行うためにサーバ・ネット・コードがDMAエンジンを用いるということを示すために直接データ転送ルーチンによっても用いられる。好ましい実施例では、チェックサム・オペレーションが同期であるので、単一のTIBが用いられる。

FLEXPOOLは、メモリ管理に対して空間を割り当てる。

【0028】図6は、どのように二つのサーバ・ネットIDが、ディスク及びそのミラー・ディスクへの二つの経路に対して用いられるかを示している。好ましい実施例では、二つのサーバ・ネットIDは、SCSIチップ40及び42に関連付けられる。図6に示すように、これらのSCSIコントローラ・チップ40及び42は、ディスク100及びそのミラー・ディスク115へのアクセスを供給する。好ましい実施例では、フォールト・フリー（無欠陥）システムは、次のものを備えている：CPU 020及びCPU122；二つのサーバ・ネットID 10及び12；ディスク及びそのミラー・ディスク14への1次（主要）経路及びディスク及びそのミラー・ディスク16へのバックアップ経路；サーバ・ネットアダプタ30及び32；SCSIチップ40、42、44及び46；ディスク100及びそのミラー・ディスク115；チェーン120及び122。この構成では、システムの全てのコンポーネントは、バックアップとして少なくとも一つの冗長コンポーネントを有する（例えば、ディスク100は、バックアップ・ミラー・ディスク115を有する）。従って、データ書き込みオペレーションの間中、全てのデータ転送は、エラーが発生しかつ正しいデータが一つのディスクからアクセスすることができないならば、そのデータもまた、アクセスのために別のディスクに配置されるように二つのディスク（ディスク及びそのミラー・ディスク）に対してなされる。この構成は、もしあれば、データの汚染を結果としてほとんど生じない。ディスク読取りオペレーションの間中、直接データ転送ソフトウェアは、ディスク100または115の一つのみによってアクセスのためにデータ・バッファをマップする。それゆえに、バッファは、一つのサーバ・ネット仮想アドレス空間の中にだけマップされる。

【0029】代替実施例では、4つのサーバ・ネットIDが直接データ転送のために用いられる。この構成では、データ・バッファは、全ての4つのSCSIコントローラ・チップ40、42、44及び46によるアクセスに対するサーバ・ネット仮想アドレス空間の中に冗長的にマップされなければならない。これは、ディスク処理にSCSIコントローラ・チップ40、42、44または46に対してデータ転送要求を発行させる。直接データ転送支援ソフトウェアは、ソフトウェア・ルーチンのライブラリとして構成される（ルーチンと呼ばれる）。好ましい実施例では、ファイル・システムは、これらルーチンへのクライアントである。それゆえに、ファイル・システムが直接データ転送が起動されることを決定したときに、それは、バッファをマップしかつチェックサム計算を行うために必要な動作を起こすルーチンを出す。直接データ転送ルーチンが行われたときに、ファイル・システムは、メッセージをディスク処理を含んでいる適切なCPUに送るためにメッセージ・システムを用いる。

【0030】図7は、直接データ転送を有するシステム・ソフトウェア階層化の一例である。図7に示すように、ファイル・システム210は、直接メッセージ・システム230を呼出すことができかつメッセージ・システムを呼出す前に直接データ転送ライブラリ220のルーチンを出すことができる。ファイル・システム210は、フラグ・イネープリング直接データ転送が設定されたときに直接データ転送ライブラリ220を呼出す。先に示したように、このフラグは、SETMODE 141またはBULKREAD/BULKWRITEが実行されるときにセットされる。直接データ転送がFILE\_CLOSE 時間でイネールされたならば、ファイル・システムは、直接データ転送セッションを終了するために直接データ転送ルーチンを出す。これは、直接データ転送ルーチンに、サーバ・ネット・コードと二つのサーバ・ネットIDとを分解させる。FILE\_CLOSE 時間は、これ以上のファイルが転送されないことを示す。

【0031】本発明の別の実施例では、同じCPUにおける複数の処理が同じディスクまたはテープに対して直接データ転送を行っているときに、デバイス・ハンドル・キャッシング・スキームが用いられる。これは、CPUにおける同じサーバ・ネットIDを指し示すデバイス・ハンドルの複製を回避する。デバイス・ハンドル・キャッシング・システムが用いられるときには、直接データ転送ルーチンは、直接データ転送セッションが確立されたときにデバイス・ハンドルがCPUに存在するかどうかを調べるためにチェックする。デバイス・ハンドルが既に存在しているならば、そのデバイス・ハンドルが用いられる。さもなければ、新しいデバイス・ハンドルがサーバ・ネット・コードを呼出すことによって生成される。FILE\_CLOSE 時間では、デバイス・ハンドルは、除去されない。

【0032】読取りデータ転送オペレーションに対して、チェックサム及び妥当性検査は、データ転送の終了後に行われる。好ましい実施例では、直接データ転送起動ルーチンは、(1)チェックサム・バッファを割り当て、(2)サーバ・ネット仮想アドレス空間の中に要求プロセッサのバッファをマップし、かつ(3)ファイル・システムにリターンする。マッピングは、サーバ・ネット・アダプタにサーバ・ネットの1次（主要）経路にわたりデータをプッシュさせるために行われる。このマッピング・オペレーションは、要求CPUのメモリがプロセッサのキャッシュと一貫しているようにするプロセッサ・キャッシュ・スイープを含む。ファイル・システムは、マップされたサーバ・ネット仮想アドレスをその要求制御領域に組込む。次いで、関連メッセージは、適切な媒体サーバ処理（ディスク処理またはテープ処理）に送られる。要求処理からのこのメッセージは、転送が直接データ転送であることを示す。次いで、ディスク処理は、応答データ・バッファ及びチェックサム・バ

ッファは、サーバ・ネット・アダプタによるアクセスに対してマップされることを知る。ディスク処理は、

(1) どのオペレーションを実行すべきか及び(2) 指定されたアドレスにおける宛先CPU (要求処理を含んでいるCPU) に読取りデータ及びチェックサムを配置すること、をそれに報告するサーバ・ネット・アダプタに指令を発行する。サーバ・ネット・アダプタは、データ転送が終了したときにディスク処理を含んでいるCPUに割込む。この構成は、図4に示されている。

【0033】好ましい実施例では、サーバ・ネット仮想アドレス空間の中への要求プロセッサのバッファのマップは、直接データ転送ルーチンによって行われる。これらのサーバ・ネット仮想アドレスは、ディスク処理へ転送される。この転送を容易にするために、直接データ転送ルーチンは、それがディスク処理を含んでいるCPUに送る要求にサーバ・ネット仮想アドレスを組込むファイル・システムにサーバ・ネット仮想アドレスをリターンする。データ転送が終了したときに、サーバ・ネット・アダプタは、要求が行う割込みを介してディスク処理を通知する。要求処理が、データ転送終了したというメッセージを受け取ったときに、それは、読取り要求の終了を処理するために直接データ転送ルーチンを呼出す。次いで、呼出された直接データ転送ルーチンは、要求CPUのバッファに対する適切なチェックサムを計算するためにサーバ・ネット・コードを呼出す。このチェックサム・オペレーションは、ブロッキング・オペレーションである。従って、処理は、チェックサム・オペレーションが処理の間、中断される。ルーチンは、チェックサム計算が終了したときに呼出し処理をアンブロックする(ブロックしない)。次いで、直接データ転送ルーチンは、計算されたチェックサムがサーバ・ネット・アダプタによってリターンされたチェックサムと一致することを確認する。次いで、ルーチンは、呼出しファイル・システム・ルーチンへ成功またはチェックサム・エラー指示のいずれかをリターンする。

【0034】図8は、直接データ転送を有する読取り要求経路での処理フローを示す。段階300では、アプリケーションは、読取り要求に対するファイル・システムを呼出す。段階302では、ファイル・システムは、要求セクションを準備しかつ直接データ転送ルーチンを呼出す。段階304では、直接読取り起動ルーチンは、FLEXPOOLからのチェックサム・バッファを割り当てる。段階306では、直接読取り起動ルーチンは、サーバ・ネットにわたりサーバ・ネット・アダプタへのバッファのアクセスをイネーブルするために要求CPUのバッファをマップし、かつサーバ・ネット仮想アドレスをファイル・システムにリターンする。段階308では、ファイル・システムは、ディスク処理へメッセージを送るためにメッセージ・システムを呼出す。このメッセージは、要求セクションに組込まれるバッファのサーバ・ネット

仮想アドレスを含む。段階310では、ディスク処理は、サーバ・ネット・アダプタへワーク要求を発行する。このワーク要求は、読取りデータ及びチェックサムを、要求処理を含んでいるCPUに配置するためのものである。段階312では、サーバ・ネット・アダプタは、データ転送を実行しそして終了通知でディスク処理に割込む。段階314では、ディスク処理は、要求処理のメッセージに応答する。

【0035】図9は、直接データ転送を伴う読取り応答経路に対する処理フローを示す。段階320では、ディスク処理の応答は、要求処理のメッセージをキューにさせ、かつ要求処理が起動される。段階322では、要求処理が起動しかつチェックサムを確認する。段階324では、直接読取り終了ルーチンは、DMAエンジンに対するチェックサム計算をキューするためにサーバ・ネット・コードを呼出す。次いで、呼出し処理を中断する。DMAエンジンは、次いで、チェックサム計算を実行する。段階326では、DMAエンジン・チェックサム計算終了割込みは、サーバ・ネット・コードに要求処理をアンブロックさせる。段階328では、直接データ転送ルーチンは、サーバ・ネット・アダプタによってデジョット(配置)されたチェックサムで計算されたチェックサムを確認する。段階330では、直接データ転送ルーチンは、FLEXPOOLにチェックサム・バッファをリターンしかつファイル・システムに対するチェックサム比較の成功または失敗をリターンする。段階332では、好ましい実施例では、ファイル・システムは、比較が成功であれば、要求処理へリターンするか、または、チェックサム・エラーが発生したならば、直接データ転送なしで再度試みる。

【0036】書込みデータ転送オペレーションに対して、チェックサムは、データが転送される前に計算される。ファイル・システムは、直接書込み起動ルーチンを呼出しかつ要求CPUのバッファのアドレスを供給する。直接データ転送ルーチンは、チェックサム・バッファをまず割り当てかつ関連バッファ(要求データ及びチェックサム・バッファ)をサーバ・ネット仮想アドレス空間の中にマップする。書込みオペレーションに対して、バッファは、ディスク及びそのミラーの二つの経路によりアクセスに対してサーバ・ネット仮想アドレス空間の中にマップされる。これがミラーされた書込みであれば、両方の関連サーバ・ネット・アダプタは、これらのバッファをアクセスする。ミラーされた書込みは、好ましい実施例で用いられる。

【0037】次いで、直接データ転送ルーチンは、バッファのチェックサムを計算するためにサーバ・ネット・コードを呼出す。サーバ・ネット・コードは、要求CPUのDMAエンジンにチェックサム計算を次いでキューしかつチェックサム計算の終了まで呼出し処理をブロックする。DMAエンジンは、指定したチェックサムを計

算しかつ合成チェックサムをチェックサム・バッファに配置する。チェックサム計算に対する終了割り込みは、サーバ・ネット・コードによって次いでフィールドされる。終了割り込みは、その時に要求処理をアンブロックする。直接データ転送ルーチンは、ファイル・システムに4つのサーバ・ネット仮想アドレスをリターンすることによって処理を再開する。これらのサーバ・ネット仮想アドレスの二つは、要求CPUのバッファに対応する。他の二つのサーバ・ネット仮想アドレスは、チェックサム・バッファに対応する。ファイル・システム書き込みルーチンは、これらの4つのサーバ・ネット仮想アドレスを、そのメッセージの要求制御セクションの中に組み込みそして関連メッセージをディスク処理へ送るためにメッセージ・システムを呼出す。好ましい実施例では、データ汚染を回避するために、要求処理バッファは、ディスク読取りが実行されるときに一つのサーバ・ネット・アダプタにだけマップされる。

【0038】ディスク処理は、この直接データ転送書き込み要求の受け取りにより、SCSIコントローラへ適切なワーク要求を送る。このワーク要求は、データ及びチェックサム・バッファが要求処理を含んでいるCPUから来ることを指定するが、サーバ・ネット・アダプタからの要求終了割り込みは、ディスク処理を含んでいるCPUに行くべきであることを指定する。次いで、サーバ・ネット・アダプタは、要求処理のCPUから書き込みオペレーションに対するデータをプルする。次いで、データは、物理的媒体に書き込まれ、そしてディスク処理は、終了により割り込みを介して通知される。ディスク処理は、1次及びミラー・ディスクの半分の両方からの終了割り込みを待つ。両方のディスクがオペレーションを終了した後、ディスク処理は、要求処理のメッセージに応答する。

【0039】ディスク処理によって送られた応答が要求処理のCPUに到着したとき、CPUは、要求処理を起動する。次いで、ファイル・システムは、書き込み終了を処理するために直接データ転送ルーチンを呼出す。この場合には、直接データ転送ルーチンは、サーバ・ネット仮想アドレス空間からのバッファをアンマップ(unmaps)し、チェックサム・バッファの割り当てを解除し、かつファイル・システムにリターンする。次いで、ファイル・システムは、そのユーザに書き込み要求の終了を通知する。図10は、直接データ転送を伴う書き込み要求経路に対する処理フローを示す。段階350では、アプリケーションは、書き込みオペレーションに対するファイル・システムを呼出す。段階352では、ファイル・システムは、直接データ転送書き込み起動ルーチンを呼出す。段階354では、直接データ転送書き込み起動ルーチンは、FLEXPOOLからのチェックサム・バッファを割り当てる。段階356では、直接データ転送書き込み起動ルーチンは、二つのコントローラ経路のよるアクセスに対する要求C

PUのバッファ及びチェックサム・バッファをマップする。次いで、ルーチンは、チェックサム計算を行うためにサーバ・ネット・コードを呼出す。段階358では、サーバ・ネット・コードは、チェックサム計算をDMAエンジンにキューしかつ要求処理を中断する。段階360では、終了割り込みは、サーバ・ネット・コードに要求処理を再開させかつ直接書き込み起動ルーチンにリターンさせる。段階362では、直接データ転送書き込み起動は、要求及びチェックサム・バッファのサーバ・ネット仮想アドレスをファイル・システムにリターンする。段階364では、ファイル・システムは、これらのサーバ・ネット仮想アドレスをその要求制御領域に組み込みかつこのメッセージをディスク処理へ送る。

【0040】図11は、直接データ転送を伴う書き込み応答経路に対する処理フローを示す。段階370では、コードは、要求処理を起動する。段階372では、ファイル・システムは、直接データ転送書き込み終了ルーチンを呼出す。段階374では、この直接データ転送ルーチンは、サーバ・ネット仮想アドレス空間からのバッファをアンマップし、チェックサム・バッファの割り当てを解除してリターンする。段階376では、ファイル・システムは、アプリケーションに書き込み終了通知をリターンする。好ましい実施例では、チェックサム・バッファに対する空間は、直接データ転送ライブラリ・ルーチンによって割り当てられる。この空間は、FLEXPOOLから割り当てられる。代替実施例では、ファイル・システムは、チェックサム・バッファに対する空間を割り当てる。チェックサム・バッファ・サイズは、要求CPUのバッファ・サイズ及びセクタ/ブロック・サイズに依存する。

【0041】好ましい実施例では、直接データ転送オペレーションを起動するファイル・システム・クライアントは、また、そのオペレーションを取り消すこともできる。この取消しは、データ転送の終りまたは顕著なデータ転送要求が存在する間に行うことができる。この方法で直接データ転送オペレーションが取り消されると、ファイル・システムは、取消しを実行するために直接データ転送ルーチンを呼出す。このルーチンは、サーバ・ネット仮想アドレス空間からのバッファをアンマップすることに対しかつ関連チェックサム・バッファをそれらのプールにリターンすることに対して責任がある。ファイル・システムは、次いで、取消し通知を適切なディスク処理へ送る。これら二つのオペレーションが終了した後、直接データ転送オペレーションが取り消されと考えられる。直接データ転送ルーチンがバッファをアンマップするとき、サーバ・ネット・アダプタは、それがこのバッファをアクセスすることを試みるならばエラーに遭遇する。これらのエラーは、サーバ・ネット・アダプタ及び要求処理を含んでいるCPUの両方に報告される。好ましい実施例では、サーバ・ネット・アダプタは、第1のエラーに遭遇した後バッファをアクセスする

ことを停止する。

【0042】好ましい実施例では、システムにおける全てのCPUのサーバ・ネットワークIDは、ファームウェアで登録される。この登録は、ファームウェアにおける“セット・サーバ・ネットワーク・パラメータ”メールボックス指令の形である。このメールボックス指令に応じて、ファームウェアは、8ビット・サーバ・ネットワーク・ハンドルをリターンする。このハンドルは、ファームウェアに対するホスト・プロセッサを識別するためにSCSIモジュール・ドライバによって用いられる。このサーバ・ネットワークIDの登録は、要求プロセッサ（要求処理を含むプロセッサ）と記憶装置（ディスクを含む装置）との間で直接データ転送を許容する。上記で説明した例は、要求処理及びディスク処理が異なるCPUに配置される場合だけを考慮した。これら二つの処理が同じCPUに配置されるならば、直接データ転送は、まだ用いることができる。この例で直接データ転送が用いられるときに、要求処理とディスク処理との間のバッファのCPUコピーは、まだ回避される。従って、また、直接データ転送は、要求処理及びディスク処理の両方が同じCPUに配置されるときにも望ましい。

【0043】上記の例では、ディスクは、記憶デバイスとして用いられたが、直接データ転送に対してあらゆる記憶デバイスを用いることができる。例えば、直接データ転送は、不必要なデータ・コピーを回避するためにテ\*

\*ープI/Oに用いることができる。そのI/Oがチェックサム計算を含まないならば、直接データ転送は、ディスクの場合よりも簡単である。テープ処理が直接データ転送も支持するならば、バックアップ及び復元施設は、この特徴を利用することができる。例えば、ディスク処理の場合、テープ処理及びバックアップ/復元処理は、3つの異なるCPUに全て分散され、直接データ転送は、二つの不必要なバッファ・コピーを回避しうる。好ましい実施例では、次に示すソフトウェア・コードの抜粋は、直接データ転送ライブラリにおけるインターフェイス・ルーチンを供給する。第1の抜粋は、直接データ転送セッション開始ルーチン（このパラグラフの真下を参照）に対するものである。このルーチンは、それが直接データ転送要求を支持することができるディスク処理からの指示をそれが受け取るときにファイル・システムによって呼出される。ディスク処理は、この指示を有するSETMODE 141/バルク・データ転送要求に応答する。ディスク処理からの応答は、また、サーバ・ネットワークID、パケットサイズ型、等のような、直接データ転送ルーチンによって用いられる情報を含む。この情報は、ファイル・システムによって直接データ転送セッション開始ルーチンにパスされる。

【0044】  
【表1】

```

/*
** Returns 0 upon success
**      an error code upon failure
** iop_cookie is an input pointer to the area that the IOP returned to the
**      File System to pass on to the Direct IO routines.
**      The File System should not care
**      about the internal format of this area. The format of this area would be
**      an agreement between the IOPs and the Direct IO routines. This area
**      will contain the Tnet IDs of the controller, the type of the packetizer
**      in the controller, etc.
**
** directio_cookie is a pointer to an area where directio can store some
**      of its context for the directio session. Aside from allocating storage for
**      this area, the File System should not care about the format of this area.
**
** The size of the cookie areas is TBD.
*/
int DirectIO_Session_Start(void *iop_cookie, void *directio_cookie);
{
    look at iop cookie and see how many tnet ids are in it.
    call tser_dev_install to install these tnet ids. Ask tser_dev_install not to
    worry about allocating interrupt and barrier AVTs.
    call tser_dev_set_packetizer to set the packetizer to the type specified
    in the iop_cookie.
    call tser_dev_set_tnetid to set the tnet id in the installed device handle.
    store the returned device handles in the directio cookie area.
    allocate a Tib from FLEXPPOOL and store the Tib address in the
    directio_cookie area.
}

```

直接データ転送セッション終了ルーチンは、FILE\_CLOS 能にするために実行されるときにファイル・システムに  
E 時間でまたはSETMODE 141 が大きいデータ転送を不 50 よって呼出される。



[0045]

\* \* [表2]

```

/*
 * Returns 0 upon success
 * an error code upon failure.
 *
 * directio_cookie is a pointer to the area that was initialized during the
 * directio_session_start call.
 */
int DirectIO_Session_End(void *directio_cookie)
{
    call tser_dev_remove to uninstall the Tnet IDs.
    deallocate the TIB buffer by returning it to the FLEXPPOOL.
}

```

直接読取り開始ルーチンは、直接データ転送読取り要求 ※ [0046]  
 を起動する前にファイル・システムによって呼出され 【表3】  
 る。 ※

```

/*
 * Returns 0 upon success
 * an error code upon failure.
 *
 * directio_cookie is a pointer to the direct io session information that was
 * initialized during directio session start.
 * buffer is a pointer to the buffer into which data should be read.
 * buffer_size is the size of the above buffer.
 * tnet_vaddrs is a pointer to an array of two 32 bit integers. Two tnet
 * virtual addresses will be deposited there by Direct_Read_Start.
 * These two tnet vaddrs then need to be copied to the Request Control
 * and sent on to the IOP.
 * xsum_buffer is a pointer to a 32 bit integer location. Direct_Read_Start
 * deposits the address of the xsum buffer it allocated in this location.
 * This address needs to be passed to Direct_Read_End or
 * Direct_Read_Abort.
 */
int Direct_Read_Start(void *directio_cookie, void *buffer, int buffer_size,
                     void *tnet_vaddrs, void *xsum_buffer)
{
    *xsum_buffer = allocate the checksum buffer from the Flex Pool.
    map the user buffer and the checksum buffer for write access by the
    device handle stored in the directio_cookie.
    deposit the two tnet virtual addresses returned by the map routine into the
    area pointed at by tnet_vaddrs.
}

```

直接読取り終了ルーチンは、それがディスク処理から直  
 接データ転送読取り要求に対する応答を受け取るときに  
 ファイル・システムによって呼出される。ディスク処理 40  
 応答が成功を示すならば、直接読取り終了ルーチンは、  
 チェックサム妥当性検査を行う。全ての場合に、直接読

取り終了ルーチンは、バッファをアンマップしかつチェ  
 ックサム・バッファの割り当てを解除する。  
 [0047]  
 【表4】

```

Returns 0 upon success
an error code upon failure. One such failure could be checksum mismatch.

tnet_vaddr is a pointer to the two tnet virtual addresses that were created
during the call to Direct_read_start.
iop_returned_status is the status code that the IOP returned in its response to
the read request sent by the File System.
xsum_buffer is the 32 bit address of the checksum buffer that Direct_Read_Start
allocated.
*/
int Direct_Read_End(void *directio_cookie, void *tnet_vaddr, int iop_returned_status,
void xsum_buffer)
{
unmap the two tnet virtual addresses listed in the tnet_vaddr array.

if (iop_returned_status == success)
{
use the TIB from the directio cookie area.
call tser_transfer to calculate checksum.
compare device returned checksum to calculated checksum.
set return value.
}
return xsum_buffer to FLEXPOOL.
return return_value
}

```

直接書き込み開始ルーチンは、直接データ転送書き込み要求 \* [0048]  
を起動する前にファイル・システムによって呼出され \* [表5]  
る。

```

Returns 0 upon success.
otherwise an error code is returned.

directio_cookie is a pointer to the area of memory that belongs to Direct IO
that is allocated by the File System.

buffer is a pointer to the user buffer that is being written to the media.
buffer_size is the size of the above buffer.
tnet_vaddr is a pointer to an array of four 32 bit integers. The array needs to
be allocated by the file system. Direct IO routines will deposit four Tnet
virtual addresses into this array - two for the xsum buffer and two for the user
buffer. The FS needs to copy these four into the request control it sends to
the IOP.

xsum_buffer is a pointer to a 32 bit location where Direct_Write_Start will
deposit the address of the allocated xsum buffer. This address needs
to be passed to Direct_Write_End or Direct_Write_Abort.
*/
int Direct_Write_Start(void *directio_cookie, void *buffer, int buffer_size,
void *tnet_vaddr, void *xsum_buffer)
{
*xsum_buffer = allocate a xsum buffer from the FLEXPOOL.
call the Tnet millicode map routines to map the xsum and user buffers for
access by the disk and its mirror.
deposit the Tnet virtual addresses returned above into tnet_vaddr array.
use the TIB whose address we stored in the directio_cookie.
call tser_transfer to calculate checksum of buffer.
}

```

直接書き込み終了ルーチンは、直接データ転送書き込み要求 \* [0049]  
への応答がディスク処理から受け取られるときにファイ \* [表6]  
ル・システムによって呼出される。



テムは、チェックサム計算を実行することに対して責任がある。代替実施例では、チェックサム計算は、ファイル・システム・コードで実行することができる。この例では、サーバ・ネット・コードが非同期インターフェイスだけを供給するので、割込みハンドラは、チェックサム計算の終了割込みを処理するために用いられる。別の実施例では、サーバ・ネット・コードは、同期インターフェイスを供給する。この構成では、チェックサムは、割込みハンドラなしでファイル・システム・レイヤにおいて実行することができる。

【0052】本発明の全てかつ完全な開示がなされると同時に、種々の代替及び変更が、特許請求の範囲の範疇から逸脱することなく本発明の種々の態様に対してなされうることが当業者に明らかになるであろう。

【0053】

【発明の効果】本発明のデータ処理システムは、データを転送するデータ処理システムであって、要求CPUを含んでいる複数の中央処理装置（CPUs）；CPUsの一つが記憶装置へのアクセスを制御する少なくとも一つの記憶装置；複数のCPUsと記憶装置とを相互接続するネットワーク；要求CPUによる記憶装置の直接アクセスを供給し、要求CPUのバッファに対する仮想メモリ・アドレスを生成しかつ記憶装置アクセス要求と共に仮想メモリ・アドレスを、記憶装置へのアクセスを制御しているCPUsの一つに供給する要求CPUにおける手段、ワーク要求に回答しておりかつネットワークを通してCPUsの一つと直接インターフェイスしている記憶装置に仮想メモリ・アドレスを含んでいるワーク要求を送るCPUsの一つにおける手段、を含んでいるアクセス手段；を備え、データは、要求CPUと記憶装置との間で直接転送されるので、CPUsと記憶装置との間のデータの転送中に類似するデータ・コピーを除去することができる。

【図面の簡単な説明】

【図1】記憶装置アーキテクチャの一例を示す図であ \*

\*る。

【図2】ディスク転送の一例を示す図である。

【図3】コントローラが“プッシング”しているときに発生するデータの転送を示す図である。

【図4】ディスク読取りに対する直接データ転送の一例を示す図である。

【図5】ディスク書き込みに対する直接データ転送の一例を示す図である。

10 【図6】どのように二つのサーバ・ネットIDがディスク及びそのミラー・ディスクへの二つの経路に対して用いられるかを示す図である。

【図7】直接データ転送を伴うシステム・ソフトウェア階層化の一例である。

【図8】直接データ転送を伴う読取り要求経路を有する処理フローを示す図である。

【図9】直接データ転送を伴う読取り応答経路に対する処理フローを示す図である。

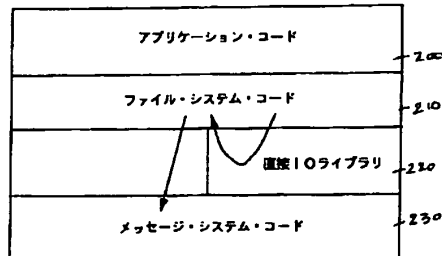
【図10】直接データ転送を伴う書き込み要求経路に対する処理フローを示す図である。

20 【図11】直接データ転送を伴う書き込み応答経路に対する処理フローを示す図である。

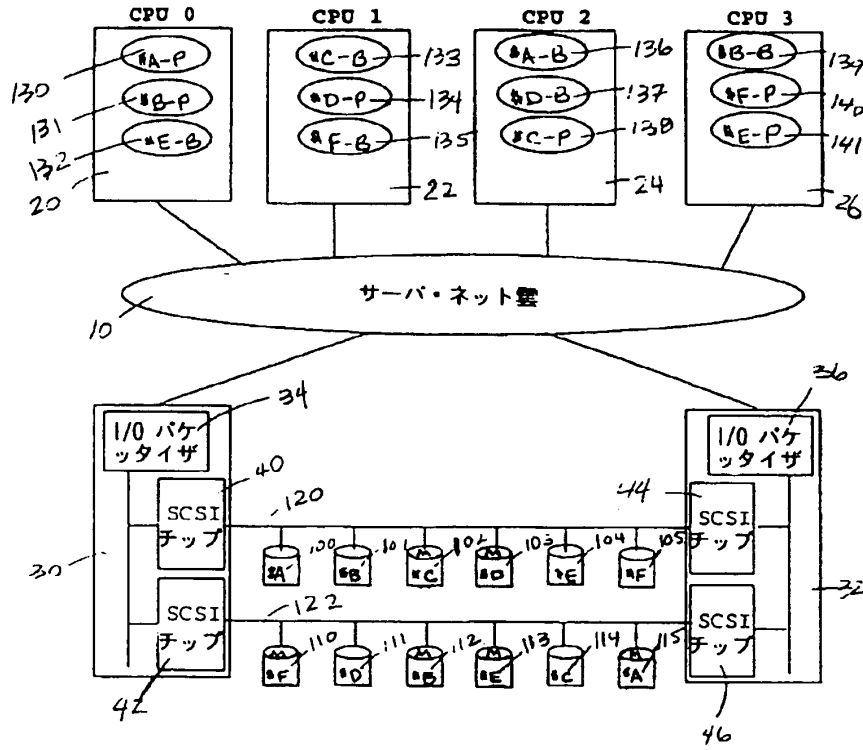
【符号の説明】

- 10 サーバ・ネット雲
- 26 CPU3
- 30 サーバ・ネット・アダプタ
- 34 I/Oパッケージ
- 40 SCSIチップ
- 105 ディスク
- 120 SCSIチェーン
- 30 162 バッファ
- 170 データ
- 172 サーバ・ネット仮想アドレス
- 174 ソース・ノード
- 176 宛先ノード

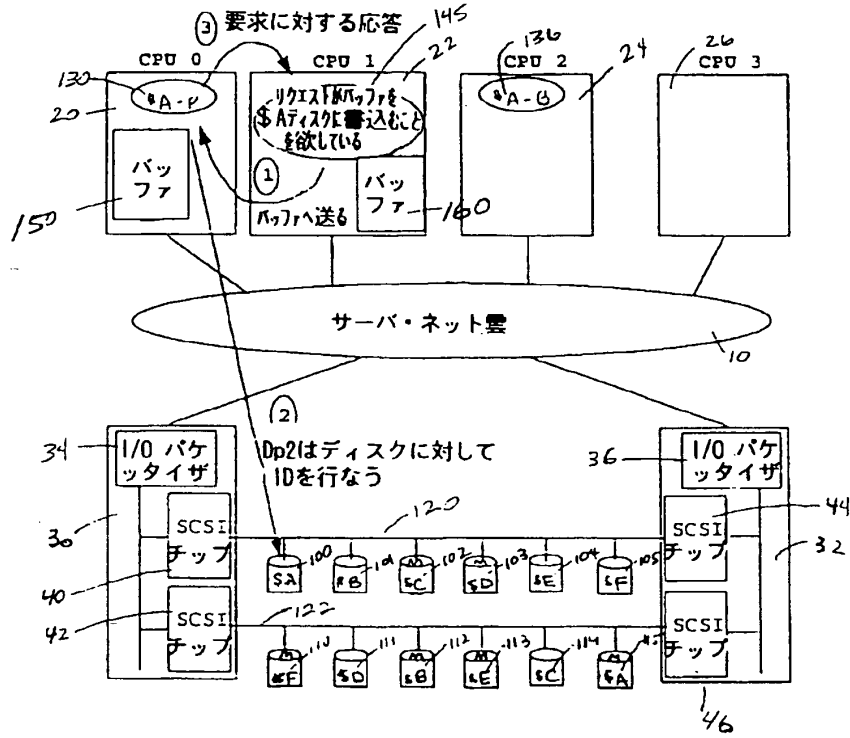
【図7】



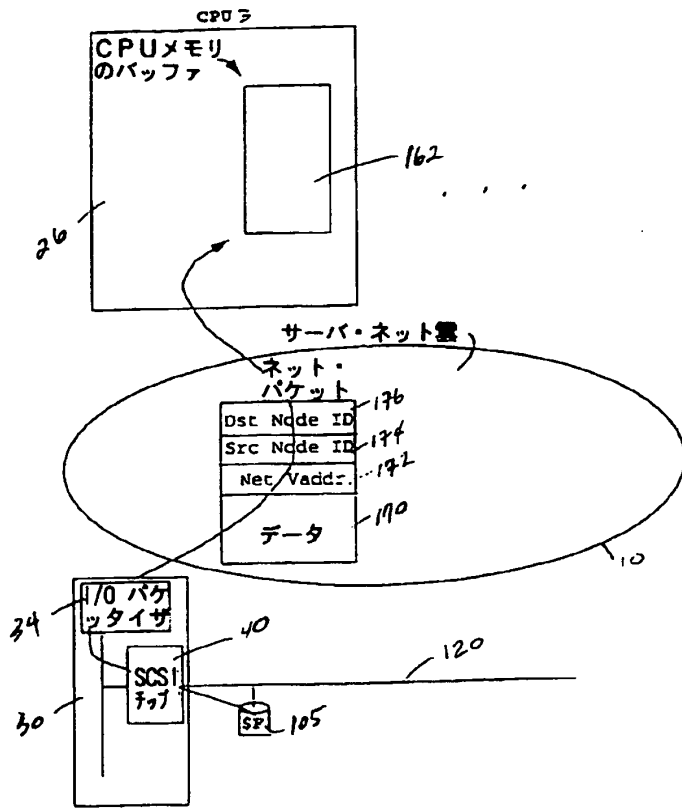
【図1】



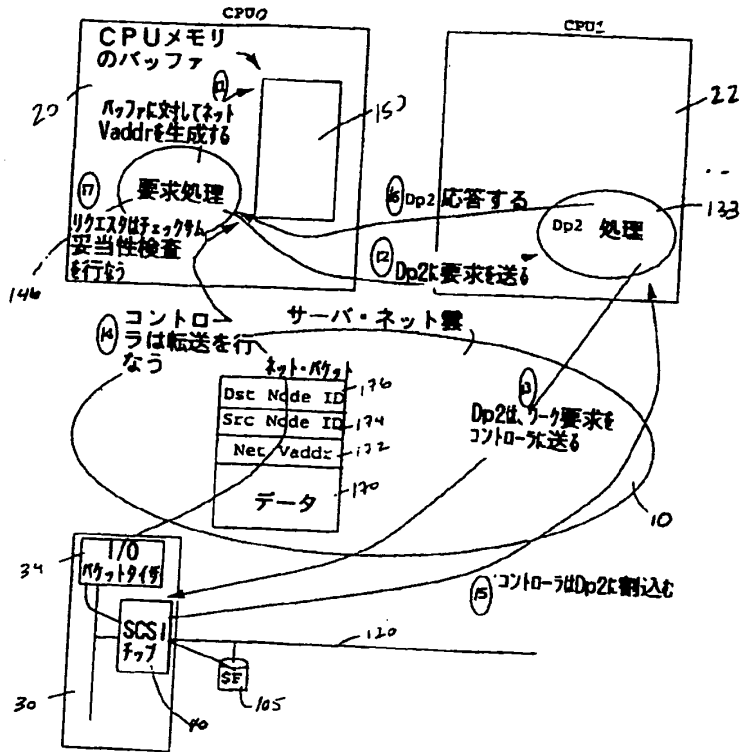
【図2】



【図3】

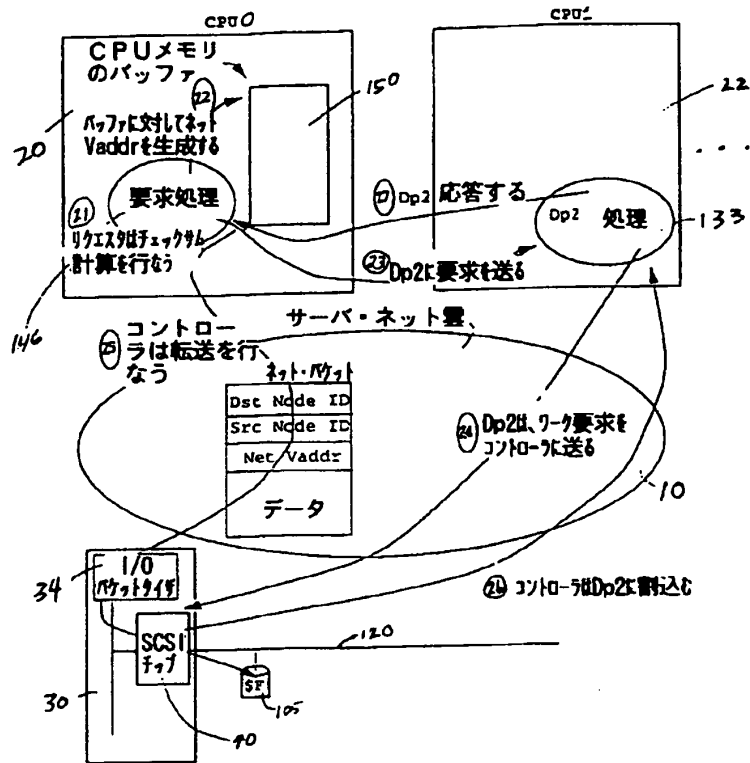


【図4】

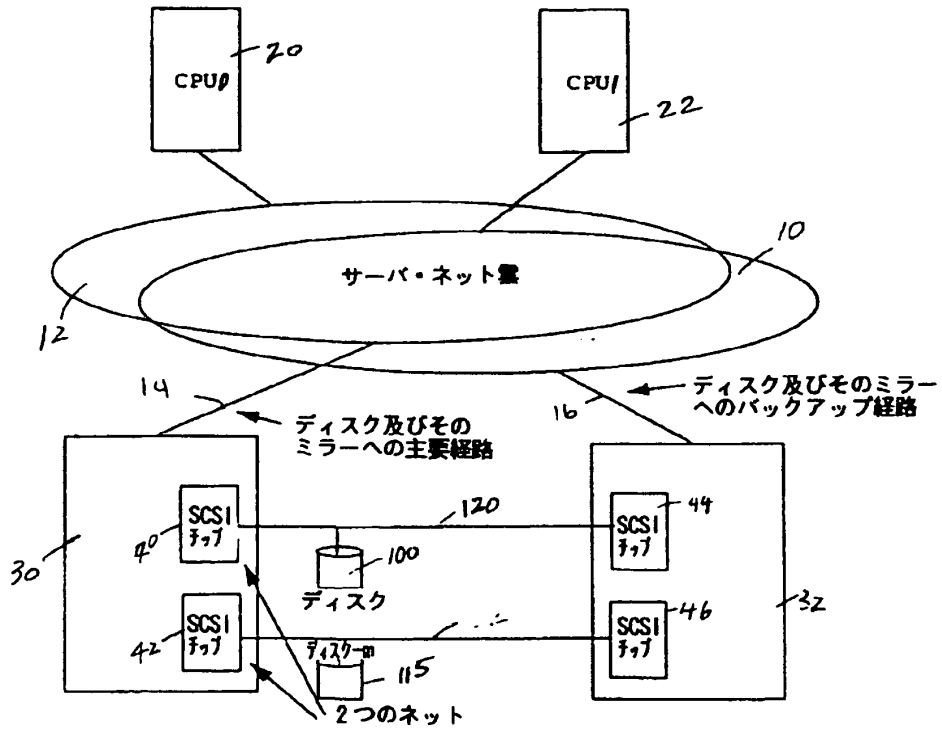




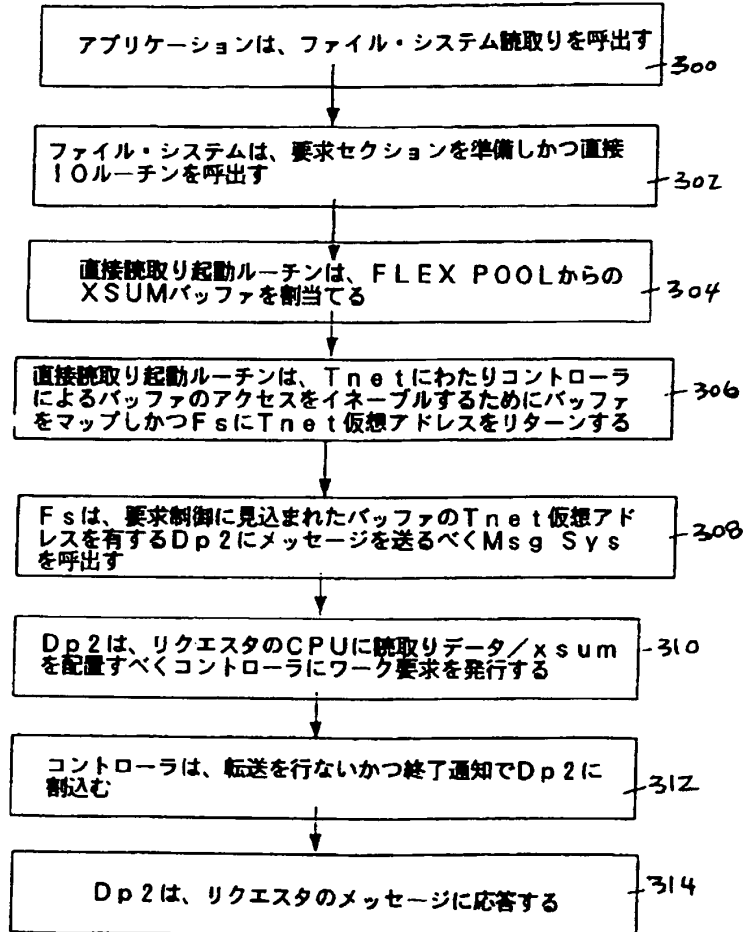
【図5】



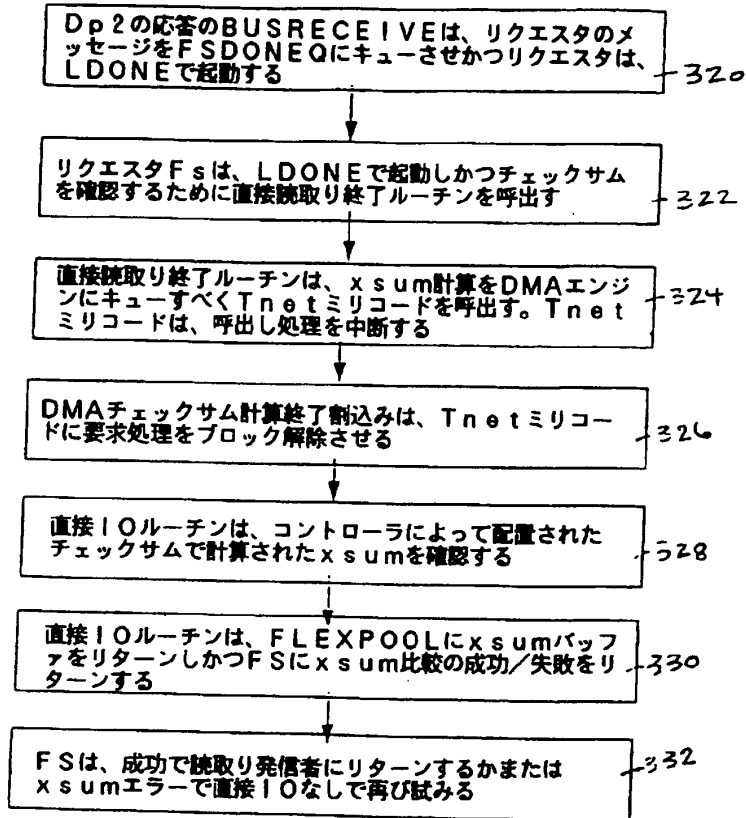
【図6】



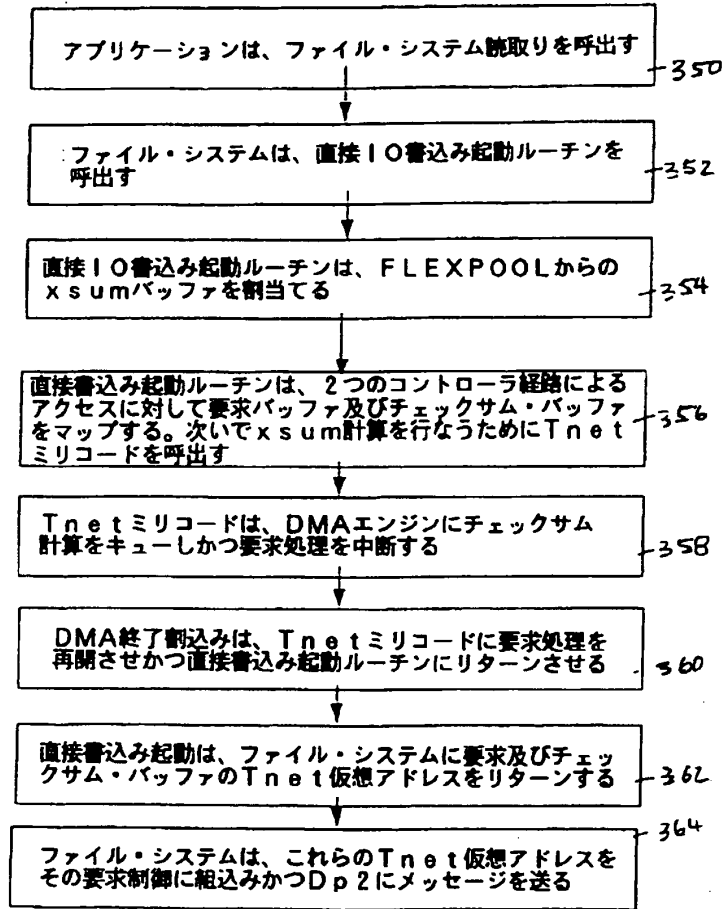
【図8】



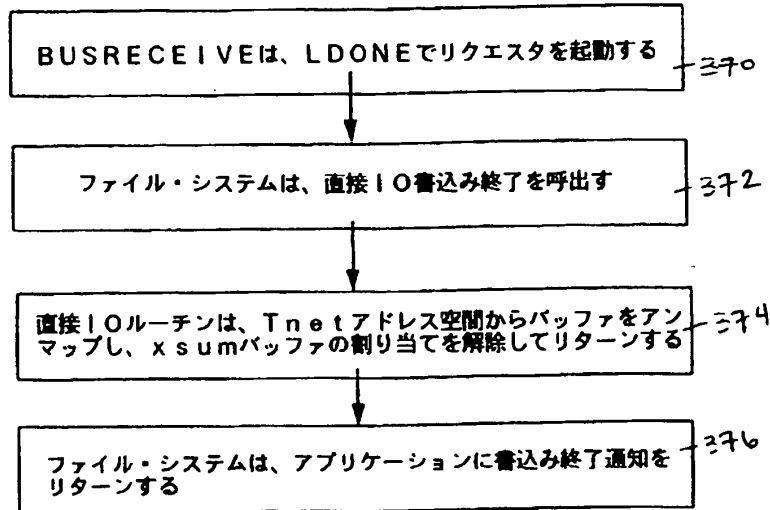
【図9】



【図10】



【図11】



フロントページの続き

(72)発明者 スリニヴァサ ディー マーシー  
アメリカ合衆国 カリフォルニア州  
95131 サン ホセ ゴールデンレイク  
ロード 1410

(72)発明者 アニル カートリ  
アメリカ合衆国 カリフォルニア州  
95148 サン ホセ ヘリテージ エステ  
ータス ドライヴ 3281

【公報種別】 特許法第 17 条の 2 の規定による補正の掲載  
【部門区分】 第 6 部門第 3 区分  
【発行日】 平成 16 年 11 月 11 日 (2004.11.11)

【公開番号】 特開平 9-185594  
【公開日】 平成 9 年 7 月 15 日 (1997.7.15)  
【出願番号】 特願平 8-301770  
【国際特許分類第 7 版】  
G 0 6 F 15/163

【F I】  
G 0 6 F 15/16 3 2 0 R  
G 0 6 F 15/16 3 2 0 V

【手続補正書】  
【提出日】 平成 15 年 11 月 12 日 (2003.11.12)  
【手続補正 1】  
【補正対象書類名】 明細書  
【補正対象項目名】 特許請求の範囲  
【補正方法】 変更  
【補正の内容】  
【特許請求の範囲】  
【請求項 1】

複数の中央処理装置 (CPU s) 及び少なくとも一つの記憶装置がネットワークによって相互接続される、前記 CPU s が要求処理を有している要求 CPU を含んでおり、前記記憶装置へのアクセスが前記 CPU s の一つによって制御される、データを転送するデータ処理システムにおいて、前記記憶装置と前記要求 CPU との間でのデータの直接転送の方法であって、

- a) 前記要求処理が前記要求 CPU のメモリ・バッファを表す仮想メモリ・アドレスを生成する段階と、
  - b) 前記要求処理が前記仮想メモリ・アドレス及び記憶装置アクセス要求を前記 CPU s の前記一つに送る段階と、
  - c) 前記 CPU s の一つが前記仮想メモリ・アドレスを含んでいるワーク要求を前記記憶装置に送る段階と、
  - d) 前記ワーク要求に応答する段階と、及び
  - e) 前記記憶装置と前記メモリ・バッファとの間のデータの転送に対して前記要求 CPU と直接インタフェースする段階と
- を具備することを特徴とする方法。

【請求項 2】  
処理システムの複数のデータ・プロセッサ装置の一つと記憶装置との間でデータを転送する方法であり、前記複数のデータ・プロセッサの前記一つがメモリ記憶素子及びデータ転送を要求する要求処理を含んでおり、前記システムが前記データ転送を始動するために前記複数のデータ・プロセッサ装置で実行される制御処理を有する、該方法であって、前記メモリ記憶素子における位置を表す仮想メモリ・アドレスを含んでいる、データ転送要求を前記要求処理から前記制御処理に送る段階と、前記仮想メモリ・アドレスを前記制御処理から前記記憶素子に転送する段階と、前記記憶素子の制御下で前記仮想メモリ・アドレスを用いて前記記憶素子と前記メモリ記憶装置における位置との間でデータを転送する段階を具備することを特徴とする方法。

【請求項 3】  
記憶メモリを有している一つのプロセッサ装置を含む、複数のプロセッサ装置と、

記憶装置と、  
その間でデータを転送するために前記複数のプロセッサ装置と前記記憶装置とを相互接続する通信媒体と、  
前記メモリ位置を表す仮想メモリ・アドレスを含むデータ転送要求を送ることによって前記記憶装置と前記記憶メモリにおけるメモリ位置との間でデータ転送を要求するために前記一つのプロセッサ装置で実行される要求処理と、及び  
前記仮想メモリ・アドレスを前記記憶素子に供給することによって前記記憶素子と前記メモリ位置との間の前記データ転送を始動するために前記データ転送要求を受取りかつ応答する制御処理と  
を備えていることを特徴とする処理システム。



JP1995020994A

1995-1-24

BAB  
BQD

**Bibliographic Fields**

**Document Identity**

(19)【発行国】	(19) [Publication Office]
日本国特許庁 (JP)	Japan Patent Office (JP)
(12)【公報種別】	(12) [Kind of Document]
公開特許公報 (A)	Unexamined Patent Publication (A)
(11)【公開番号】	(11) [Publication Number of Unexamined Application]
特開平7-20994	Japan Unexamined Patent Publication Hei 7- 20994
(43)【公開日】	(43) [Publication Date of Unexamined Application]
平成7年(1995)1月24日	1995 (1995) January 24*

**Public Availability**

(43)【公開日】	(43) [Publication Date of Unexamined Application]
平成7年(1995)1月24日	1995 (1995) January 24*

**Technical**

(54)【発明の名称】	(54) [Title of Invention]
記憶システム	STORAGE SYSTEM
(51)【国際特許分類第6版】	(51) [International Patent Classification, 6th Edition]
G06F 3/06 301 B	G06F3/06301B
12/08 320 7608-5B	12/083207608-5B
13/12 330 T 8133-5B	13/12330T8133-5B
【請求項の数】	[Number of Claims]
9	9
【出願形態】	[Form of Application]
OL	OL
【全頁数】	[Number of Pages in Document]
21	21

**Filing**

【審査請求】	[Request for Examination]
未請求	Unrequested
(21)【出願番号】	(21) [Application Number]
特願平5-162021	Japan Patent Application Hei 5- 162021
(22)【出願日】	(22) [Application Date]
平成5年(1993)6月30日	1993 (1993) June 30 days

JP1995020994A

1995-1-24

**Parties**

**Applicants**

(71)【出願人】

【識別番号】

000005108

【氏名又は名称】

株式会社日立製作所

【住所又は居所】

東京都千代田区神田駿河台四丁目6番地

(71) [Applicant]

[Identification Number]

000005108

[Name]

HITACHI LTD. (DB 69-054-1503)

[Address]

Tokyo Chiyoda-ku Kanda Surugadai 4-Chome 6

**Inventors**

(72)【発明者】

【氏名】

二宮 龍也

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72) [Inventor]

[Name]

Ninomiya Tatsuya

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address  
Hitachi Ltd. (DB 69-054-1503) storage unit systems  
department \*

(72)【発明者】

【氏名】

増崎 秀文

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72) [Inventor]

[Name]

\*\*Hidefumi

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address  
Hitachi Ltd. (DB 69-054-1503) storage unit systems  
department \*

(72)【発明者】

【氏名】

黒沢 弘幸

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72) [Inventor]

[Name]

Kurosawa Hiroyuki

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address  
Hitachi Ltd. (DB 69-054-1503) storage unit systems  
department \*

(72)【発明者】

【氏名】

高橋 直也

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72) [Inventor]

[Name]

Takahashi Naoya

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address  
Hitachi Ltd. (DB 69-054-1503) storage unit systems  
department \*

JP1995020994A

1995-1-24

(72)【発明者】

【氏名】

井上 靖雄

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72)【発明者】

【氏名】

岩崎 秀彦

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72)【発明者】

【氏名】

星野 政行

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

(72)【発明者】

【氏名】

磯野 聡一

【住所又は居所】

神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内

Agents

(74)【代理人】

【弁理士】

【氏名又は名称】

武 顕次郎

Abstract

(57)【要約】

【目的】

大形計算機の記憶システムで、システム規模を容易に拡張変更でき、システムの縮退及び活線挿抜による保守を可能とする。

(72) [Inventor]

[Name]

Inoue Yasuo

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address Hitachi Ltd. (DB 69-054-1503) storage unit systems department \*

(72) [Inventor]

[Name]

Iwasaki Hidehiko

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address Hitachi Ltd. (DB 69-054-1503) storage unit systems department \*

(72) [Inventor]

[Name]

Hoshino Masayuki

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address Hitachi Ltd. (DB 69-054-1503) storage unit systems department \*

(72) [Inventor]

[Name]

Isono Satoshi \*

[Address]

Kanagawa Prefecture Odawara City Kouzu 2880address Hitachi Ltd. (DB 69-054-1503) storage unit systems department \*

(74) [Attorney(s) Representing All Applicants]

[Patent Attorney]

[Name]

Takeshi Kenjiro

(57) [Abstract ]

[Objective ]

With storage system of large scale computer , be able to expand easily and be able to modify system scale , conservation is made possible with degeneracy and livewire \*

挿抜による保守を可能とする。

【構成】

上位 CPU と接続される複数のホストアダプタ(上位側インタフェース)1 と、アレイディスク 5 と接続される複数のディスクアダプタ(記憶装置側インタフェース)2 と、これらのアダプタに共用される一時記憶用キャッシュメモリ 3 とは、これらアダプタ及びキャッシュメモリに共用される共通バス 4 上に挿抜自在に取り付けられる。

規模を拡大するには、必要な数だけこれらアダプタ 1,2 及びキャッシュメモリ 3 を付加するだけでよい。

アダプタ 1,2, キャッシュメモリ及び共通バスは二重化され、障害時の縮退運転を可能とし、また各アダプタ及びキャッシュメモリと共通バスとの結合部は、活線挿抜可能としシステム無停止で保守点検部品交換を可能とする。

removal of system .

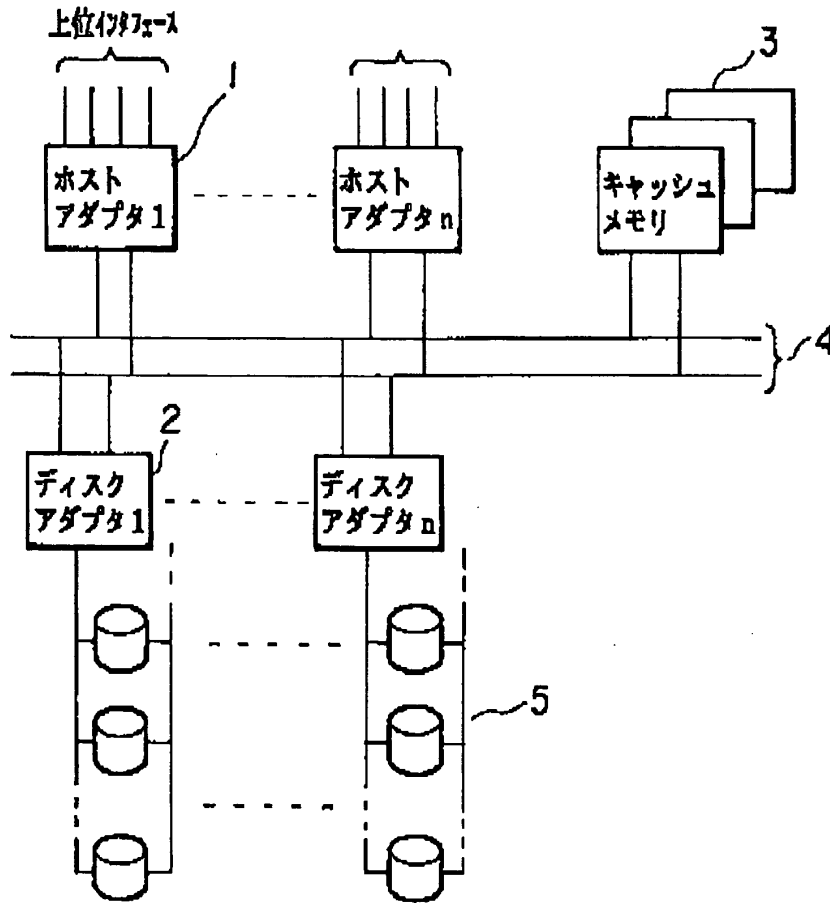
[Constitution ]

host adapter of plural which upper position CPU is connected (upper side interface ) disk adapter of plural which 1 and array disk 5 is connected (storage device side interface ) temporarily cache memory 3 for storage which is shared in 2 and these adapter , \* removal it is installed unrestrictedly in these adapter and on common bus 4 which is shared in cache memory .

scale is expanded, just required number just to add these adapter 1, 2 and the cache memory 3 has.

adapter 1, 2, cache memory and common bus are done, doubling make degeneracy driving at the time of damage possible, in addition bonding section of each adapter and cache memory and common bus makes live wire \* removal possible and makes preservation inspection parts exchange possible in system non stop.

【図1】



Claims

【特許請求の範囲】

[Claim (s)]

【請求項 1】

[Claim 1]

上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置及び前記複数の記憶装置側接続論理装置間で転送されるデータを一時記憶するキャッシュメモリ装置とを有する記憶システムにおいて、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、これらの装置に共用される共通バスにより相互に接続されるように構成したことを特徴とする記憶システム。

## 【請求項 2】

前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、いずれもモジュールで構成し、前記モジュールは、それぞれ、前記共通バスに対し挿抜自在に取付けられるように構成したことを特徴とする請求項 1 記載の記憶システム。

## 【請求項 3】

前記共通バスは、プラッタ上に配設され、前記上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記プラッタに対し挿抜自在に取付けられるように構成したことを特徴とする請求項 1 または 2 記載の記憶システム。

## 【請求項 4】

前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記共通バスは、いずれも少なくとも二重化されており、前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記共通バスの一方により縮退運転が可能となるように構成したことを特徴とする請求項 1 ないし 3 のいずれか 1 記載の記憶システム。

## 【請求項 5】

前記二重化された上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置は、

storage device side connected logic device of plural which configuration does the upper side connected logic device of plural which configuration does interface for upper position device and interface for storage device and aforementioned storage device and, In storage system which possesses cache memory device which remembers upper side connected logic device of aforementioned plural and data which is transferred between storage device side connected logic device of the aforementioned plural at one time, as for storage device side connected logic device, and aforementioned cache memory device of upper side connected logic device, aforementioned plural of aforementioned plural, Way it is connected mutually, by common bus which is shared in these device configuration storage system. which designates that it does as feature

## [Claim 2]

Way storage device side connected logic device, and aforementioned cache memory device of the upper side connected logic device, aforementioned plural of aforementioned plural configuration do in each case with module, as for aforementioned module, respectively, vis-a-vis aforementioned common bus \* removal unrestrictedly are installed, configuration storage system. which is stated in Claim 1 which designates thing which is done as feature

## [Claim 3]

Aforementioned common bus was arranged on [puratta], storage device side connected logic device, of aforementioned upper side connected logic device, aforementioned plural and each of module which configuration does the aforementioned cache memory device, vis-a-vis aforementioned [puratta] \* removal that unrestrictedly it is installed, stated in Claim 1 or 2 which designates that configuration it does as feature storage system.

## [Claim 4]

Aforementioned upper side connected logic device, aforementioned storage device side connected logic device, aforementioned cache memory device, and aforementioned common bus are done in each case doubling at least, Way degeneracy driving becomes possible, aforementioned upper side connected logic device, aforementioned storage device side connected logic device, aforementioned cache memory device, and aforementioned common bus depending upon on one hand configuration the storage system. which is stated in any 1 of Claims 1 through 3 which designates the thing which is done as feature

## [Claim 5]

As for upper side connected logic device, storage device side connected logic device, cache memory device which

いずれも活線挿抜ができるように構成したことを特徴とする請求項 4 記載の記憶システム。

**【請求項 6】**

前記記憶装置は、二重化された電源部を備えたことを特徴とする請求項 1 ないし 6 のいずれか 1 記載の記憶システム。

**【請求項 7】**

前記記憶装置は、複数の小形記憶装置を組み合わせたアレイ記憶装置で構成したことを特徴とする請求項 1 ないし 3 のいずれか 1 記載の記憶システム。

**【請求項 8】**

前記キャッシュメモリ装置は、キャッシュメモリ本体を持ち、前記共通バスに直接取り付けられるキャッシュメモリモジュールと、キャッシュメモリを持つ増設用のキャッシュユニットとを有しており、前記キャッシュユニットは、前記共通バスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して接続されるように構成したことを特徴とする請求項 1~7 のいずれか 1 記載の記憶システム。

**【請求項 9】**

前記上位側接続論理装置及び前記記憶装置側接続論理装置は、それぞれ、二重化されたマイクロプロセッサを有し、両マイクロプロセッサによりデータの比較チェックを行なうように構成したことを特徴とする請求項 1 ないし 8 のいずれか 1 記載の記憶システム。

**Specification**

**【発明の詳細な説明】**

**【0001】**

**【産業上の利用分野】**

本発明は、大形計算機システムやネットワークシステム等に接続される磁気ディスク装置、磁気テープ装置、半導体記憶装置、または光ディスク装置等の記憶装置を制御する記憶制御装置を含む記憶システムに係り、特に、システムの拡張性が高く縮退運転や活線挿抜対応の可能な記憶システムに関する。

**【0002】**

theaforementioned doubling is done, which way live wire \* removal is possible, configuration storage system . which is stated in Claim 4 which designates thing which is done as feature

**[Claim 6 ]**

As for aforementioned storage device , doubling storage system . which is stated in any 1 of Claims 1 through 6 which designates that it has power supply section which is done as feature

**[Claim 7 ]**

As for aforementioned storage device , with array storage device which combines the small shape storage device of plural configuration storage system . which is stated in any 1 of Claims 1 through 3 which designates that it does as feature

**[Claim 8 ]**

As for aforementioned cache memory device , we have possessed cache unit for addition which has cache memory module and cache memory which are directly installed in aforementioned common bus with cache memory main body , as for theaforementioned cache unit , Through cache port package for addition which , \* removal is installed directly unrestrictedly in aforementioned common bus way it is disconnected, configuration storage system . which is stated in any 1 of the Claim 1 ~7 which designates thing which is done as feature

**[Claim 9 ]**

Way aforementioned upper side connected logic device and aforementioned storage device side connected logic device have microprocessor which respectively, the doubling is done, do relative check of data with both microprocessor , the configuration storage system . which is stated in any 1 of Claim 1 to 8 which designates thing which is done as feature

**[Description of the Invention ]**

**[0001]**

**[Field of Industrial Application ]**

this invention relates to storage system which includes storage controller which controls magnetic disk device , magnetic tape device , semiconductor storage device , or optical disk device or other storage device which is connected to large scale computer system and network system etc, especially, extendibility of system regards possible storage system of degeneracy driving and live wire \* removal correspondence highly.

**[0002]**

## 【従来の技術】

従来、大形計算機に接続される記憶システムとして、例えば特開昭 61-43742 号公報に記載されているように、上位装置(CPU)に対するインタフェース(ホストアダプタ)、キャッシュメモリ、及び磁気ディスク装置等の記憶装置に対するインタフェース(ディスクアダプタ)の相互間をホットライン(専用線)で接続しているものが知られている。

## 【0003】

図 20 は、従来の記憶システムの構成の概要を示す図である。

同図において、201-1~201-n はそれぞれ複数の上位ホスト(CPU)に接続されるホストアダプタ(対上位論理モジュール)、202-1~202-n は、共有の大形ディスク装置 205 に接続されるディスクアダプタ(記憶媒体接続用論理モジュール)、203 は、複数のホストアダプタに共有のキャッシュメモリ、206 は同様に共有の管理メモリである。

従来装置では、各ホストアダプタ 201-1~201-n とキャッシュメモリ 203 の間、キャッシュメモリ 203 と各ディスクアダプタ 202-1~202-n の間、各ホストアダプタ 201-1~201-n と管理メモリ 206 の間、並びに管理メモリ 206 と各ディスクアダプタ 201-2~201-n の間は、それぞれ別々のホットライン 207-1~207-n 及び 208-1~208-n によって接続されている。

また、これらのホストアダプタ及びディスクアダプタの監視及び保守を行なう保守用プロセッサ(SVP、図示せず)も各々のホストアダプタ及びディスクアダプタにそれぞれ専用線を介して接続されている。

## 【0004】

## 【発明が解決しようとする課題】

上記従来技術では、上位装置に対するホストアダプタ(対上位接続論理モジュール)と、記憶装置に対するディスクアダプタ(対記憶媒体接続論理モジュール)と、キャッシュメモリ(キャッシュメモリモジュール)との各間がホットラインで接続されているため、装置構成が複雑になると共に、ホストアダプタ、キャッシュメモリ、ディスクアダプタ、ディスク装置等、装置としての拡張性に乏しく、いわゆるスケラブル(拡張及び縮小自在)なシステム構成が得られなかった。

また、システムを多重化することにより障害発生時等に縮退運転(2 台のうち 1 台を停止し他の 1 台だけで運転するなど)や活線挿抜対応(システムを動作したままで基板や回路の部品等挿し

## [Prior Art]

Until recently, as stated in for example Japan Unexamined Patent Publication Showa 61-43742 disclosure as storage system which is connected to large scale computer, interface for upper position device (CPU) (host adapter), those which connect between mutual of interface (disk adapter) for cache memory, and magnetic disk device or other storage device with the hot line (private line) are known.

## [0003]

Figure 20 is figure which shows gist of configuration of the conventional storage system.

In same Figure, as for 201 - 1 - 201 n host adapter which is connected to upper position host (CPU) of respective plural (Anti- upper position logic module), as for 202 - 1 - 202-n, disk adapter which is connected to large scale disk drive 205 of joint ownership (logic module for storage media connection), as for 203, as for cache memory, 206 of joint ownership it is a management memory of joint ownership in same way in host adapter of plural.

Until recently with device, each host adapter 201-1~201-n and between cache memory 203, the cache memory 203 and between each disk adapter 202-1~202-n, between each host adapter 201-1~201-n and management memory 206, and as for management memory 206 and between each disk adapter 201-2~201-n, the respective separate hot line 207-1~207-n and it is connected with 208 - 1 - 208-n.

In addition, also these host adapter and processor (SVP, not shown) for conservation watch the disk adapter and conservation through private line to each host adapter and disk adapter respectively, it is connected.

## [0004]

## [Problems to be Solved by the Invention]

Because host adapter for upper position device (Anti- upper position connected logic module) with, disk adapter for storage device (Anti- storage media connected logic module) with, cache memory (cache memory module) with each between is connected with hot line, as equipment configuration becomes complicated, generally known scaleable (Extended and reduction unrestricted) system configuration could not acquire with above-mentioned Prior Art, scantily in the extendibility, as device such as host adapter, cache memory, disk adapter, disk drive.

In addition, degeneracy driving (It stops among 1 2 and drives at just other 1 such as) and what makes live wire \* removal corresponding (system it is operated to put substrate and part etc of the circuit such as you can apply with while)



かえるなど)を可能とすることがなにも配慮されておらず、このため、障害発生時の部品交換やシステムの制御プログラムをグレードアップするときには、システムを一時停止し対応しなければならぬ等の問題があった。

[0005]

従って、本発明の目的は、上記従来技術の問題点を解決し、コモンバス方式を採用することにより、システム構成(規模)に応じてホストアダプタ、記憶装置アダプタ等の各論理モジュールやキャッシュメモリ及び記憶媒体を接続することでスケラブルなシステムを実現することができるようにすると共に、各論理モジュール、記憶媒体及びコモンバスの多重化により、縮退運転と各論理モジュール及び記憶媒体の活線挿抜対応とを可能とし、無停止で保守することができる記憶システムを提供することにある。

[0006]

【課題を解決するための手段】

上記目的を達成するため、本発明は、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置及び前記複数の記憶装置側接続論理装置間で転送されるデータを一時記憶するキャッシュメモリ装置とを有する記憶システムにおいて、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、これらの装置に共用されるコモンバスにより相互に接続されるように構成する。

[0007]

前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、いずれもモジュールで構成し、前記モジュールは、それぞれ、前記コモンバスに対し挿抜自在に取付けられるように構成する。

[0008]

前記コモンバスは、プラッタ上に配設され、前記上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記プラッタに

possible nothing being considered at time etc of damage by multiplexing doing system , because of this , when the parts exchange at time of damage and grade up doing control program of the system , it must stop system at one time and must correspond therewas a or other problem .

[0005]

Therefore, as objective of this invention solves problem of theabove-mentioned Prior Art , can be actualized requires scaleable system by the host adapter , storage device adapter or other each logic module and fact that cache memory and storage media areconnected by adopting common bus system , according to system configuration (scale ) , in each logic module , storage media and multiplexing of common bus depending, It is to offer storage system which degeneracy driving and makes each logic module andlive wire \* removal correspondence of storage media possible, the conservation it is possible in non stop.

[0006]

[Means to Solve the Problems ]

In order to achieve above-mentioned objective , as for this invention, the storage device side connected logic device of plural which configuration does upper side connected logic device of plural which configuration does interface for the upper position device and interface for storage device and aforementioned storage device and, In storage system which possesses cache memory device which remembers upper side connected logic device of aforementioned plural and data which istransferred between storage device side connected logic device of theaforementioned plural at one time, as for storage device side connected logic device , and aforementioned cache memory device of upper side connected logic device , aforementioned plural of aforementioned plural , In order to be connected mutually, by common bus which is shared inthese device configuration it does.

[0007]

storage device side connected logic device , and aforementioned cache memory device of the upper side connected logic device , aforementioned plural of aforementioned plural configuration do in each case with module , aforementioned module ,respectively, vis-a-vis aforementioned common bus \* removal in orderunrestrictedly to be installed, configuration do.

[0008]

Aforementioned common bus is arranged on [puratta ] , storage device sideconnected logic device , of aforementioned upper side connected logic device , aforementioned plural and each of module which configuration does theaforementioned

対し挿抜自在に取付けられるように構成する。

[0009]

前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記共通バスは、いずれも少なくとも二重化されており、前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記共通バスの一方により縮退運転が可能となるように構成する。

[0010]

前記二重化された上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置は、いずれも活線挿抜ができるように構成する。

[0011]

前記記憶装置についても、同様に二重化された電源部を備えることができる。

[0012]

前記記憶装置は、複数の小形記憶装置を組み合わせたアレイ記憶装置で構成することができる。

[0013]

前記キャッシュメモリ装置は、キャッシュメモリ本体を持ち、前記共通バスに直接取り付けられるキャッシュメモリモジュールと、キャッシュメモリを持つ増設用のキャッシュユニットとを有しており、前記キャッシュユニットは、前記共通バスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して接続されるように構成することができる。

[0014]

前記上位側接続論理装置及び前記記憶装置側接続論理装置は、それぞれ、二重化されたマイクロプロセッサを有し、両マイクロプロセッサによりデータの比較チェックを行なうように構成することができる。

[0015]

なお、共通バス上には、上記上位側接続論理モジュールとは別の形式の上位側インタフェースや、上記記憶装置側接続論理モジュールとは別の形式の記憶装置側インタフェースを置き換

cache memory device , vis-a-vis aforementioned [puratta ] \* removal in order unrestrictedly to be installed, configuration does.

[0009]

A aforementioned upper side connected logic device , aforementioned storage device side connected logic device , aforementioned cache memory device , and aforementioned common bus are done in each case doubling at least, in order for degeneracy driving to become possible, aforementioned upper side connected logic device , aforementioned storage device side connected logic device , aforementioned cache memory device , and the aforementioned common bus depending upon on one hand configuration .

[0010]

As for upper side connected logic device , storage device side connected logic device , cache memory device which the aforementioned doubling is done, which in order for live wire \* removal to be possible, configuration it does.

[0011]

Concerning aforementioned storage device , in same way it is possible to have power supply section which doubling is done.

[0012]

A aforementioned storage device configuration is possible with array storage device which combines small shape storage device of plural .

[0013]

As for aforementioned cache memory device , we have possessed cache unit for the addition which has cache memory module and cache memory which are directly installed in aforementioned common bus with cache memory main body , as for the aforementioned cache unit , through cache port package for addition which, \*removal is installed directly unrestrictedly in aforementioned common bus in order to be connected, configuration we are possible.

[0014]

A aforementioned upper side connected logic device and aforementioned storage device side connected logic device have microprocessor which respectively, doubling is done, in order to do relative check of data , with both microprocessor configuration is possible.

[0015]

Furthermore, it is possible also, to replace on common bus and/or to increase storage device side interface of another form from upper side interface and the above-mentioned storage device side connected logic module of another form from

えたり増設したりすることもできる。

[0016]

【作用】

上記構成に基づく作用を説明する。

[0017]

本発明によれば、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、これらの装置間で転送されるデータを一時記憶するキャッシュメモリ装置(複数の上位側接続論理装置及び複数の記憶装置側接続論理装置に共有されるキャッシュメモリ装置)とを有する記憶システムにおいて、前記複数の上位装置側接続論理装置、複数の記憶側接続論理装置、及びキャッシュメモリ装置は、これらの装置に共有される共通バスにより相互に接続されるように構成したので、上位側接続論理装置と記憶装置側接続論理装置とキャッシュメモリの増設または変更は、単にこれらを共通バス上に追加しまたは変更して行くだけでよく、増設によるアップグレードが容易に達成できスケラブルなシステム構成を得ることができる。

[0018]

また、これらの上位側接続論理装置、記憶装置側接続論理装置及びキャッシュメモリ装置は、モジュール化されて、共通バスの配設されたブラケットに挿抜(着脱)自在に取り付けるようにしたので、これらの装置の必要な数量の増設作業も簡単である。

[0019]

また、上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置、及びこれらの間を接続する共通バスは、二重化され、2系統に分けて配線されているので。

これらの装置の一方に障害が発生したときでも、他方の装置を用いて縮退運転が可能である。

なお、障害発生時に縮退運転状況を示す情報は、共有メモリに書き込まれる。

[0020]

この場合、上位側接続論理装置、記憶装置側接

theabove-mentioned upper side connected logic module .

[0016]

[Working Principle ]

Action which is based on above-mentioned configuration is explained.

[0017]

According to this invention , in storage system which possesses cache memory device (upper side connected logic device of plural and cache memory device which is shared to storage device side connected logic device of plural )which remembers storage device side connected logic device of plural which the configuration does upper side connected logic device of plural which configuration does interface for upper position device and interface for storage device and theaforementioned storage device and data which is transferred between the device of these at one time, Because in order to be connected mutually, by common bus which isshared to these device configuration it did storage side connected logic device , and cache memory device of upper position equipment side connected logic device , plural of aforementioned plural , as for upper side connected logic device and storage device side connected logic device and addition or modification of cache memory , Simply, it can add these on common bus and or it just modifies to benecessary, can achieve up grade easily with addition and can acquire scaleable system configuration .

[0018]

In addition, modulization being done, \* removal (attachment and detachment ) you install these upper side connected logic device , storage device side connected logic device and cache memory device ,unrestrictedly in [puratta ] where common bus is arranged, because itrequired, also addition job of required number quantity of these device issimple.

[0019]

In addition, upper side connected logic device , storage device side connected logic device , cache memory device , and the common bus which connects between these to be done doubling , dividing into2 system , because wiring it is done.

When these device damage occurs on one hand even, degeneracy driving ispossible making use of device of other .

Furthermore, data which shows degeneracy operating condition at time of damage is written to joint ownership memory .

[0020]

In case of this , because upper side connected logic device ,

統論理装置、及びキャッシュメモリ装置は、いずれも活線挿抜対応のコネクタ部を具備しているため、システムを停止することなく保守点検を行なって故障部品の交換を行ったり、増設用の部品を追加したりすることが可能である。

[0021]

電源部も二重化され、それにより無停電電源装置を実現する。

[0022]

記憶装置は、複数の小形記憶装置を組み合わせたアレイ形とされ、これにより従来の大形ディスク装置1台を用いたものに比べてアクセスタイムを短縮できる。

[0023]

キャッシュメモリ装置は、コモンバスに直接取り付けられるキャッシュメモリモジュール(キャッシュメモリパッケージ)と、増設用のキャッシュユニットとで構成され、増設用のキャッシュユニットは、コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して必要数接続されるようになっているので、簡単に増減することができる。

[0024]

異常により、高信頼性の記憶システムを得ることができる。

[0025]

[実施例]

以下に、本発明の実施例を図面の図1から図18により説明する。

[0026]

図1は本発明の概念図を示す。

図1により、本実施例の概要を説明する。

[0027]

1は、対上位CPU(ホスト)接続用論理モジュールであるホストアダプタ部、2は、対記憶媒体接続用論理モジュールであるディスクアダプタ部、3は、両モジュール間で転送されるデータを一時記憶するキャッシュメモリパッケージ(キャッシュメモリモジュール)、4はホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3の間のデータ転送制御を司るコモンバス、5は、縦横にアレイ状に配置した記憶媒体である磁気ディスク群(以下「アレイディスク」という)である。

storage device side connected logic device, and cache memory device in each case possess connector section of live wire \*removal correspondence, doing preservation inspection without stopping system, it is possible to exchange breakdown part, to add part for addition.

[0021]

Also power supply section is done, doubling actualizes uninterruptible electric power supply device with that.

[0022]

storage device makes array shape which combines small shape storage device of plural, can shorten access thyme in comparison with those which because of this use conventional large scale disk drive 1.

[0023]

As for cache memory device, cache memory module which is directly installed in common bus (cache memory package) with, configuration to be done with cache unit for addition, as for the cache unit for addition, through cache port package for addition which, \*removal is installed directly unrestrictedly in common bus necessary number it is connected, because it groans, it can increase and decrease simply.

[0024]

With fault, storage system of high reliability can be acquired.

[0025]

[Working Example (s)]

Below, Working Example of this invention is explained from Figure 1 of drawing with Figure 18.

[0026]

Figure 1 shows conceptual diagram of this invention.

With Figure 1, gist of this working example is explained.

[0027]

As for 1, as for host adapter, 2 which is a logic module for anti- upper position CPU (host) connection, as for disk adapter, 3 which is a logic module for anti- storage media connection, cache memory package which remembers data which is transferred between both modules at one time (cache memory module), as for 4 as for common bus, 5 which administers data transfer control between host adapter 1, disk adapter 2, cache memory package 3, it is a magnetic disk group (Below "array disk" with you call) which is a storage media which in longitudinal and transverse is arranged in array.

ホストアダプタ 1 は、上位インタフェース側のデータ形式及びアドレス形式を記憶媒体インタフェース用のデータ形式及びアドレス形式に変換する手段と、これらを制御管理する二重化したマイクロプロセッサとを有している。

ディスクアダプタ 2 は、記憶媒体へデータを格納するためのアドレス演算機能と、記憶データ保証用冗長データの生成機能と、記憶媒体構成情報を認識する機能と、これらを制御管理するマイクロプロセッサとを有している。

[0028]

図 1 において、上位装置(CPU)から送られてきた書き込みデータは、ホストアダプタ 1 から共通バス 4 を介して一度キャッシュメモリパッケージ 3 に書き込むことにより上位に終了報告を行い、その後の空き時間でキャッシュメモリパッケージ 3 からディスクアダプタ 2 を経由してアレイディスク 5 に書き込む。

[0029]

また、上位装置からのデータ読み出し命令に対しては、キャッシュメモリパッケージ 3 上にデータが存在する場合はアレイディスク 5 からは読み出さず、キャッシュメモリパッケージ 3 上のデータを上位装置に転送する。

一方キャッシュメモリパッケージ 3 上にデータが存在しない場合は、アレイディスク 5 からディスクアダプタ 2 により共通バス 4 を経由して一度キャッシュメモリパッケージ 3 に書き込まれた後同様にホストアダプタ 1 を経由して上位装置へ転送する。

[0030]

共通バス 4 上のホストアダプタ 1、ディスクアダプタ 2、キャッシュメモリパッケージ 3 各々はその接続数を任意に変えることができる。

ホストアダプタ 1 の実装数を変えれば対上位接続バス数が増え、上位ホストに対するデータ転送能力を高めることができる。

ディスクアダプタ 2 の実装数を変えれば記憶媒体に対する接続バス数が増え、記憶媒体に対するデータの書き込み/読み出しの転送能力を高めることができる。

また、同時に記憶媒体の数も増加することができる。

キャッシュメモリパッケージ 3 の実装数を変えればデータの一時格納場所であるキャッシュメモリの容量が増え、記憶媒体の総容量に対するキャッシュメモリの容量の比率を高めることがで

host adapter 1 data form and address form of upper position interface side data form for the storage media interface and doubling which means. these which are converted to address form control is done has had microprocessor which is done.

disk adapter 2, address computing function in order to house data and generating function of the redundant data for storage data guarantee and function which recognizes storage media configuration information and, these control has had microprocessor which is done to storage media .

[0028]

In Figure 1 , writing data which is sent from upper position device (CPU ) , through the common bus 4 from host adapter 1 , does end report in upper position by writing to onetime cache memory package 3 , after that writes to array disk 5 at empty time via disk adapter 2 from cache memory package 3 .

[0029]

In addition, when data exists on cache memory package 3 vis-a-vis data reading command from upper position device , it does not read out from array disk 5 , transfers the data on cache memory package 3 to upper position device .

On one hand when data does not exist on cache memory package 3 , after being written to one time cache memory package 3 , with disk adapter 2 via common bus 4 from array disk 5 in same way it transfers to upper position device via host adapter 1 .

[0030]

To change number of connections into option it is possible host adapter 1 , disk adapter 2 , cache memory package 3 each on the common bus 4 .

If mount quantity of host adapter 1 is changed, anti- upper position connected number of passes changes, it is possible to raise data transfer capacity for upper position host .

If mount quantity of disk adapter 2 is changed, connected number of passes for the storage media changes, it is possible to raise transfer capacity of writing /reading of data for storage media .

In addition, it can increase also quantity of storage media simultaneously.

If mount quantity of cache memory package 3 is changed, data temporarily the capacity of cache memory which is a storage site to change, because it is possible, to raise ratio of capacity of cache memory for entire capacity of storage media

きるので、対上位装置からアクセスするデータがキャッシュメモリ上に存在する確率(以下「キャッシュヒット率」という)を高める等スケーラブルな装置構成を実現できる。

【0031】

図2は、図1の概念図の詳細な構成図を示したものである。

図2は、図1の複数台のホストアダプタ及び複数台のディスクアダプタのうち、それぞれ1台だけを示し、他は図示を省略している。

【0032】

ホストアダプタ1において、6はホストインターフェイスの光信号を電気信号に変換する信号変換部、7は上位データフォーマットをアレイディスク5用フォーマットに変換するフォーマット変換部である。

8はコモンバス4とのデータの授受を司るデータ転送制御部で、内部にパケット転送単位のデータを格納する記憶バッファを内蔵している。

9は活線挿抜対応可能な小振幅電流駆動形バスドライバ(以下「BTL」という)である。

【0033】

ホストからのデータ転送要求は10のマイクロプロセッサ(以下「MP」という)に引継がれ、ホストアダプタ1内のデータ転送制御は当MP10の管理下で行われる。

【0034】

MP10はMP内の障害発生を検出するなど高信頼性を確保するために2重化されており、11のチェック部で同じ動作をする2重化されたMP10とMP10'を比較チェックしている。

【0035】

12はMP10の制御プログラムを格納するブートデバイスで、このブートデバイス12には書き替え可能な大容量フラッシュメモリを採用しており、またMP10は必要に応じて13のローカルメモリに制御プログラムをコピーして使用することにより、MP10のメモリアクセス時間の高速化を実現しており、図中破線で囲まれた部分29がチャンネルアダプタモジュールであり、ホストアダプタ1には当モジュール29が2回路搭載してある。

【0036】

ディスクアダプタ2において、14はアレイディスクに書き込むデータをセクタ単位に格納するバッファメモリ、15はバッファメモリ14の制御及びデ

probability (Below "cache hit ratio" with you call ) where data which access is done existson cache memory such as is raised can actualize scaleable equipment configuration from anti-upper position device .

[0031]

Figure 2 is something which shows detailed constitution figure of conceptual diagram of Figure 1 .

Figure 2 , host adapter of plural table of Figure 1 and among disk adapter of plural table , respectively shows just 1, other things have abbreviated illustration.

[0032]

In host adapter 1, as for 6 as for signal converter , 7 which converts light signal of[hosutointaafaisu ] to electrical signal it is a format conversion section which converts upper position data format to format for array disk 5.

8 with data transfer controller which administers transfer of data of the common bus 4, has built in storage buffer which houses data of packet transfer unit in interior .

9 is live wire \* removal responsible small amplitude current drive shape bus driver (Below "BTL " with you call ) .

[0033]

data transfer request from host is taken over by microprocessor (Below "MPa " with you call ) of 10, data transfer control inside host adapter 1 is done under managing this MPa 10 .

[0034]

MPa 10 detects damage inside MPa such as in order toguarantee high reliability to double to be converted, to double whichoperates similarly in checker section of 11 MPa 10 and MPa 10\* which are converted are done relative check .

[0035]

With boot device which houses control program of MPa 10, it rewrites 12 to this boot device 12 and adopts possible large capacity flash memory , in addition, MPa 10 the copy doing control program in local memory of according to need 13, has actualized the acceleration of memory access time of MPa 10 by using, portion 29 which issurrounded with in the diagram dashed line being channel adapter module , In host adapter 1 twice road is installed this module 29.

[0036]

In disk adapter 2, as for 14 as for buffer memory , 15 which houses data whichis written to array disc in sector unit as for data control buffer , 16 which controls buffer memory 14 and

ータ転送制御を行なうデータ制御バッファ部、16 はアレイディスク 5 に書き込むデータを保証するための冗長データを生成する冗長データ生成部、17 はアレイディスク 5(ターゲット)に対するイニシエータ(SCSI のマスタ側インタフェース)である。

## 【0037】

またディスクアダプタ 2 内のデータ転送制御は、ホストアダプタ 1 と同じ構成をとる MP 周辺部(M P10,MP10', チェッカ 11、ブートデバイス 12、ローカルメモリ 13 からなりディスクアダプタ用の制御プログラムを搭載する)の管理下で行なわれる。

## 【0038】

アレイディスク 5 は、図 2 では 4 つのディスク(ターゲット)しか示していないが、実際には 1 台のディスクアダプタ 2 に対し例えば 4(横)×4(縦)~4(横)×7(縦)つのディスクで構成される。

横列は ECC グループ(ErrorCorrection Group)を構成し、各 ECC グループは例えば 3 つのデータディスクと 1 つのパリティディスクで構成される。

更に、後述のように、このようなアレイディスク 5 の 1 組に対し、二重化されたホストアダプタと二重化されたホストアダプタと二重化されたディスクアダプタを通じて、ある CPU からアクセスできるようにになっている。

そして、ホストアダプタの一方に障害が発生したときには、ホストアダプタの他方もしくはディスクアダプタの他方を通じて、同じ CPU から同じアレイディスクにアクセスすることができる。

## 【0039】

キャッシュメモリパッケージ 3 において、18 は各アダプタの MP10 が共通にアクセス可能で種々の管理情報を記憶する共有メモリ部、19 は共有メモリ制御部、20 はキャッシュメモリ部、21 はキャッシュメモリ制御部であり、両メモリ制御部 19、21 は共にメモリ書き込みデータ保証のための ECC 生成回路、読み出しデータの検査及び訂正回路を内蔵し、キャッシュメモリパッケージ 3 全体で最大 1GB のキャッシュ容量を実現しており、装置構成上は 2 面化して実装している。

## 【0040】

キャッシュメモリ容量を更に増設する場合は、キャッシュメモリパッケージ 3 の代わりに(または、キャッシュメモリパッケージ 3 に加えて)22 で示すキャッシュポートパッケージを実装し、23 で示す

controls data transfer as for redundant data generating part , 17 which forms redundant data in order to guarantee data which is written to array disk 5 it is an initiator (master side interface of SCSI ) for array disk 5 (target ).

## 【0037】

In addition data transfer control inside disk adapter 2 as host adapter 1 is done under managing MPa periphery (It consists of MPa 10, MPa 10\*, checker 11, boot device 12, local memory 13 and installs control program for disk adapter ) which takes same configuration .

## 【0038】

array disk 5 is shown with Figure 2 only disk (target ) of 4. Actually for example 4 (Side) X 4 (Length) - configuration it is done with 4 (Side) X 7 (Length) horn disk vis-a-vis disk adapter 2 of 1.

row configuration does ECC group (ErrorCorrectionGroup ), each ECC group configuration is done with parity disk of for example 3 data disk and one .

Furthermore, later mentioned way, access it is possible from a certain CPU , is done host adapter jp7 doubling which host adapter and doubling which are done via disk adapter which is done array disk 5 a this way vis-a-vis 1-set , doubling , it groans.

When and, host adapter damage occurs on one hand, from same CPU access it can make same array disk via other of host adapter or the other of disk adapter .

## 【0039】

In cache memory package 3, as for 18 as for joint ownership memory section , 19 where the MPa 10 of each adapter in common remembers various administration information with accessible as for joint ownership memory control section , 20 cache memory section, as for 21 with the cache memory controller , as for both memory control section 19, 21 together it builds in inspection and the correcting circuit of ECC producing circuit , reading data for memory writing data guarantee, actualizes cache capacity of the maximum 1GB with cache memory package 3 entirety , to 2 aspects converting, it mounts on equipment configuration .

## 【0040】

When cache memory capacity furthermore is increased, (Or, in cache memory package 3 adding ) it can mount cache port package which is shown with 22 in place of cache memory package 3, through connecting cable between [puratta ]

プラッタ(基板差し込み板)間接続ケーブルを介して 24 で示すキャッシュユニットに接続し、(すなわち、増設ユニット 24 内のキャッシュメモリには、キャッシュポートパッケージ 22 及びケーブル 23 を介してアクセスできるように構成され)、これによって、最大 8GB 2 面までキャッシュ容量を増設することができる。

図 2 では、キャッシュメモリパッケージ 2 を 2 面設けたのに加えて、キャッシュポートパッケージ 22 を実装し、これにケーブル 24 を介していくつかのキャッシュユニット 24 を接続した場合を示している。

[0041]

以上述べたホストアダプタ 1、ディスクアダプタ 2、キャッシュメモリパッケージ 3 は共通バス 4 を介してつながっているが、この共通バス中、25 は各アダプタの MP10 が共有メモリをアクセスするためのマルチプロセッサバス(以下「M バス」という)、26 は高速データ転送を行う高速 I/O バス(以下「F バス」という)である。

[0042]

高速 I/O バス 26 は通常は 64 ビット幅で 2 系統同時に動作しているが、障害発生時はどちらか 1 系統のみでの縮退動作が可能であり、また M バス 25 に障害が発生した場合は F バス 26 のどちらか 1 系統を使用して動作可能である。

[0043]

更に活線挿抜対応(挿抜の際、挿抜部品の負荷を小さくして挿抜を行なうことで、システムを稼働状態のまま挿抜を可能とする)の BTL9 を共通バス 4 のインターフェイスにすることで、ホストアダプタ 1 に障害が発生した場合、システムは自動的に本障害バスを閉塞し他のホストアダプタのバスを用いてアレイディスク 5 に対し上位(同じ CPU)からのアクセスを継続する。

保守員は、システム稼働状態において障害の発生したホストアダプタ 1 を取り除き、正常なホストアダプタ 1 をシステムに挿入し、27 の保守用プロセッサ(以下「SVP」という)から 28 の LAN を介して復旧の指示を与え、システムは交換されたホストアダプタ 1 の動作をチェックし正常であれば閉塞バスを復旧させることにより、無停止運転を実現している。

なお、図中 LANC は、LAN Controller(SVP インタフェースコントローラ)である。

SVP27 は、他のホストアダプタ及びディスクアダプタにも同様に接続され、監視及び保守が行な

(substrate insertion sheet) which is shown with 23 it can connect to cache unit which is shown with 24, (In cache memory inside namely, addition unit 24, through cache port package 22 and cable 23, access in order for it to be possible, configuration it is done), with this, can increase cache capacity to the maximum 8GB 2 aspect.

With Figure 2, although cache memory package 2 was provided 2 aspects, adding, it mounts cache port package 22, through cable 24 to this, it has shown casewhere several cache unit 24 are connected.

[0041]

Above, host adapter 1, disk adapter 2, cache memory package 3 which is expressed is connected through common bus 4, but in this common bus, as for 25 multi processor bus because MPa 10 of each adapter access does joint ownership memory (Below "Mbus" with you call), 26 is high speed I/O bus (Below "Fbus" with you call) which does high speed data transfer.

[0042]

high speed I/O bus 26 2 system operates simultaneously with usually 64 bit width, but at time of damage either one degeneracy operation with only of 1 system being possible, in addition when damage occurs in Mbus 25, either one of Fbus 26 using 1 system, it is a workable.

[0043]

Furthermore when by fact that BTL9 of live wire \*removal corresponding (At time of \* removal, making load of \* removal part small, by fact that it does \* removal, system while it is a work state \* removal possible it does) is designated as interface of common bus 4, the damage occurs in host adapter 1, system occlusion does this damage pass in the automatic and continues access from anti- upper position (Same CPU) vis-a-vis the array disk 5 making use of pass of other host adapter.

If maintenance person, removes host adapter 1 where damage occurs in system work state inserts normal host adapter 1 in system, through LAN of 28 from processor (Below "SVP" with you call) for conservation of 27, gives display of restoration, system operates host adapter 1 which is exchanged check and it is normal, in restoring occlusion pass depending, Non stop driving is actualized.

Furthermore, in the diagram LANC is LAN controller (SVP interface controller).

SVP27 is connected in same way to also other host adapter and the disk adapter, supervision and conservation are done,



われるようになっている。

【0044】

また、各アダプタの制御プログラムに変更がある場合は、SVP27 から LAN28 を介してブートデバイス 12 内にある制御プログラムの内容を書き替えることにより無停止のアップグレードが可能である。

【0045】

即ち、システムの制御プログラムをアップグレードを実施する場合は、まずホストアダプタ/ディスクアダプタの各モジュールを 1 モジュールずつ閉塞し、制御プログラムのアップグレードを行い再接続する。

以上のように 1 モジュールずつの制御プログラムの入れ換え操作を繰り返すことにより、系全体の制御プログラム入れ換えが実施される。

【0046】

図 3 は、図 2 に示した構成図に沿ってデータの流れとデータの保証を示した図である。

【0047】

上位からアレイディスクにデータを書き込む場合、例えば ESCON(光チャネルの商標名、IBM 社)から、先ず書き込み先の記憶空間上の物理アドレス情報(以下「PA」という)が送られて来た後、データ(CKD(Count Key Data)フォーマット)+CRC コードが送られてくる。

これらの光信号は信号変換部 6 で電気信号に変換すると共にパリティを生成し、フォーマット変換部 7 ではデータフォーマットを FBA(Fired Blocked Architecture)フォーマットに変換すると共に LRC(Longitudinal Redundancy Check, 長手方向冗長度チェック)コードを付加し、更に PA をデータの一部として取り込んでアレイディスク上の論理アドレス(以下「LA」という)を生成した後これら総ての情報に対してパリティを付加して Fバス 26 に送られる。

【0048】

キャッシュパッケージ 3 では、F バス 26 からのデータに対して誤り訂正可能な ECC を付加してキャッシュメモリ 20 に書き込む。

【0049】

ディスクアダプタ 2 では、F バスからのデータに対して更に CRC コードが付加され、該データ SCSI インターフェースを介してアレイディスク 5 に送られ、磁気ディスク装置個々に ECC を付加して

it groans.

【0044】

In addition, when there is modification in control program of each adapter ,through LAN 28 from SVP27, up grade of non stop is possible byrewriting content of control program which is inside boot device 12.

【0045】

Namely, control program of system case up grade is executed, each module of host adapter /disk adapter occlusion 1 module at a time it does first, does up grade of control program and reconnects.

Like above control program replacing of entire system is executed by repeatingreplacing operation of control program of 1 module at a time.

【0046】

Figure 3 is figure which shows guarantee of flow and the data of data alongside configuration diagram which is shown in Figure 2 .

【0047】

When from upper position data is written to array disk , after physical address data (Below "PA " with you call ) on storage space ahead writing is sent from for example ESCON (tradenname , IBM corporation of optical channel ) , first, the data (CKD (count key data ) format ) +CRC cord is sent.

As these light signal as it converts to electrical signal with signal converter 6, form the parity , in format conversion section 7 convert data format to FBA (FiredBlockedArchitecture ) format the LRC (Degree of LongitudinalRedundancyCheck, longitudinal direction redundand check ) cord is added, Furthermore after forming logic address (Below "LA " with you call ) on array disk , PA asportion of data taking in adding parity vis-a-vis these all data , it is sent to Fbus 26.

【0048】

With cache package 3, adding error correction possible ECC vis-a-vis data from Fbus 26, you write to cache memory 20.

【0049】

With disk adapter 2, furthermore CRC cord is added vis-a-vis data from Fbus , through said data SCSI interface \* \*, is sent by array disk 5, magnetic disk device adds ECC individually and guarantees writing data .

書き込みデータを保証している。

[0050]

アレイディスク 5 からのデータ読み出しにおいても同様に、各チェックコードを元に読み出しデータの検査/訂正を行い信頼性を高めている。

[0051]

以上のように、チェックコードはデータの長さ方向に対してはある長さ毎の水平チェック、データの垂直(幅)方向に対しては(例えばバイト単位の)垂直チェックで 2 重化されており、また転送が行われる領域間(図中一点鎖線)では当該 2 重化チェックコードのうち 1 つを必ずデータとして受け渡すことによりデータ保証に万全を期している。

[0052]

図 4 は図 1 で述べたスケラビリティを実現するための装置外観図であり、41 はアレイディスクを制御する制御ユニット部、42 はアレイディスクを実装するアレイユニット部で、本装置はこの 2 つのユニットで構成される。

[0053]

図 5 は制御ユニット 41 の実装図で(a)は正面図、(b)は側面図を表わす。

51 はホストアダプタ 1、ディスクアダプタ 2、キャッシュメモリパッケージ 3 を実装する論理架部、52 は停電時に揮発メモリであるキャッシュメモリ部に電源を供給するバッテリー部、53 はキャッシュメモリ増設時にキャッシュユニット 24 及び増設メモリ用の追加バッテリーを実装するキャッシュメモリ増設部、54 は SVP 実装部、55 は論理架に電源を供給する論理架用スイッチング電源、56 はアレイディスクの構成(容量)が小規模の場合のアレイディスク実装部、57 はアレイディスク部に電源を供給するアレイディスク用スイッチング電源、58 は両スイッチング電源 55、57 に電源を供給する商用電源制御部である。

[0054]

図 6 は大容量アレイディスクを構成するときのアレイユニット部の実装図で(a)は正面図、(b)は側面図を表わす。

[0055]

アレイディスク実装部 56 は、磁気ディスク装置を最大 112 台(8 行 x7 列 x2)実装可能であり、各磁気ディスク装置に障害が発生した場合の装置の入れ替えを容易にするために、装置の正面と背

and guarantees writing data .

[0050]

In same way, you inspect reading data on basis of each check cord regarding data reading from array disk 5 and / you correct raise reliability .

[0051]

Like above, check cord vis-a-vis vertical (width ) direction of horizontal check , data every of certain length vis-a-vis longitudinal direction of data to double is converted (for example byte unit ) with vertical check , with (in the diagram dot-dash line ) between region where in addition transfer is done expects completely safe in data guarantee inside one of this said double conversion check cord by all meansby as data transferring.

[0052]

As for Figure 4 with device external view in order to actualize [sukeerabiriti ] which isexpressed with Figure 1 , as for 41 control unit section which controls array disk , as for 42 in array unit section which mounts array disk , this device configuration is done with this 2 unit .

[0053]

As for Figure 5 as for (a ) as for front view , (b ) side view isdisplayed in mount figure of control unit 41 .

As for 51 logic rack section which mounts host adapter 1, disk adapter 2, cache memory package 3, as for 52 the battery section which supplies power supply to cache memory section which is a volatilization memory at time of electricity outage , as for 53 at time of cache memory addition cache memory addition section which mounts additional battery for cache unit 24 and addition memory , as for 54 SVP mountsection, As for 55 as for switching power supply , 56 for logic rack which supplies power supply to logic rack array disk mount section when configuration (capacity ) of the array disk is small scale , as for 57 switching power supply , 58 for array disk which supplies power supply to array disk section is commercial power supply controller which supplies power supply to both switching power supply 55, 57.

[0054]

As for Figure 6 when configuration doing large capacity array disk , as for (a ) as for front view , (b ) side view is displayed in mount figure of array unit section.

[0055]

array disk mount section 56, in order to make replacing device whenbeing maximum 112table (8 line x7 line x2 ) mount possible, damage generates magnetic disk device in each magnetic disk device easy, has taken kind of mounting

面の両面から挿抜可能となるような実装方式をとっている。

【0056】

61 はユニット全体の発熱を逃がすための冷却ファンで、冷却効果を高めると共に、騒音抑止の観点から小さな冷却ファンを使って小区分化し、床面より天井へ送風する構造をとっている。

【0057】

図7は図5で説明した論理架部の接続方式図である。

【0058】

71 はコモンバス4をプリント配線したプラッタ(基板の挿し込み用の板)であり、72 は各アダプタ、パッケージとプラッタ71を接続するためのコネクタである。

【0059】

ホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3の間のデータ転送はコモンバス4を介して行うため、各アダプタ、パッケージはコネクタ72上の任意のどの位置でも接続可能となり、ホストアダプタ1の実装数、ディスクアダプタ2の実装数を自由に変えることができる。

【0060】

一方、キャッシュ容量を増設する場合はキャッシュメモリパッケージ3をキャッシュポートパッケージ22に変えて実装するか、または図7に示すように、キャッシュメモリパッケージ3に加えてキャッシュポートパッケージ21を実装し、これに、接続ケーブル23を介してキャッシュユニット43(図2の24に相当)に接続することにより、もとの2GBの容量に加えて更に最大8GB2面分のキャッシュメモリ容量を拡張できる。

【0061】

図8は図5で示した論理架部の実装イメージ図である。

【0062】

図8で、コモンバス4は、プラッタ71上を左右方向にプリント配線されており、このプラッタ71に対して、キャッシュポートパッケージ22の基板(CP)の取付部、キャッシュメモリパッケージ3の基板(C)の取付部、ホストアダプタモジュールの基板(H)の取付部、及びディスクアダプタモジュールの基板(D)の取付部が設けられ、図の矢印84で示すように、各基板は、挿抜操作面側から着脱されるようになっていて、プラッタ71に差し込

system which becomes \* removalpossible from front face of device and both surfaces of rear surface .

【0056】

61 converts, as with cooling fan in order to let escape heating of the unit entirety , cooling effect is raised, using small cooling fan from viewpoint of noise control, to small partition from floor surface takes structure which air blowing is done to ceiling .

【0057】

Figure 7 is connection system figure of logic rack section which isexplained with Figure 5 .

【0058】

As for 71 with [puratta ] (sheet for putting being packed of substrate ) which printed circuit does common bus 4, 72 is the connector in order to connect [puratta ] 71 with each adapter , package .

【0059】

As for data transfer between host adapter 1, disk adapter 2, cache memory package 3 through common bus 4, in order to do,each adapter , package becomes connectable any position of option on connector 72, itis possible to change mount quantity of host adapter 1 and mountquantity of disk adapter 2 freely.

【0060】

On one hand, when cache capacity is increased, changing cache memory package 3 into the cache port package 22, it can mount, or as shown in Figure 7, it can mount cache port package 21 in addition to cache memory package 3, through connecting cable 23 to this, furthermore itcan expand cache memory capacity of maximum 8GB 2 surface amount by connecting to the cache unit 43 (Equal to 24 of Figure 2 ), in addition to capacity of original 2 GB .

【0061】

Figure 8 is mount image of logic rack section which is shownwith Figure 5 .

【0062】

Way with Figure 8 , common bus 4 [puratta ] on 71 printed circuit is made left and right directions ,can provide mount , of substrate (H ) of mount , host adapter module of substrate (C ) of mount , cache memory package 3 of substrate (CP ) of cache port package 22 and mount of substrate (D )of disk adapter module this [puratta ] vis-a-vis 71, shows with arrow 84 in thefigure, as for each substrate , attachment and detachment it is done from \* removal work surface side, groaning, when [puratta ] itis inserted in 71, common bus 4

まれるとコモンバス 4 と電気接続されるものである。

【0063】

81 は、ホストアダプタ 1 の基板上の下方部に実装されて、対上位インターフェイスを司る光コネクタ部、82 はディスクアダプタ 2 の基板上の下方部に実装されて、アレイディスク 5 と接続する SCSI コネクタ部、83 はキャッシュポートパッケージ 22 を実装したときの接続ケーブル 23 用の接続コネクタ部である。

85 は、キャッシュメモリパッケージ 3 の基板(C)の下方部に取付けたキャッシュメモリ本体(図 2 のキャッシュメモリ 20)である。

【0064】

各コネクタ部は、障害発生等で各アダプタ、パッケージを挿抜する際の操作性を向上させるため、接続コネクタ 83 を除き、操作面 84 側へは実装せず、ブラッタ 71 の接続側に集中実装している。

【0065】

図 9 は本発明のソフトウェア構成を示した図である。

【0066】

91 はホストアダプタ 1 のブートデバイス 12 に書き込まれるチャネルアダプタ制御プログラム(以下「CHP」という)、である。

また、ディスクアダプタ 2 のブートデバイス 12 に書き込まれるディスクアダプタ制御プログラムのうち、92 はアレイディスク固有の処理およびキャッシュメモリとアレイディスク間のデータ転送制御を受け持つディスクアダプタマスタ制御プログラム(以下「DMP」という)、93 は DMP92 の制御管理下でキャッシュメモリ 20 とアレイディスク 5 の間のデータ転送制御を受け持つディスクアダプタスレーブ制御プログラム(以下「DSP」という)である。

【0067】

ディスクアダプタ 2 のブートデバイス 12 には、DMP92 と DSP93 の 2 種類が書き込まれているが、装置構成上 n セットのディスクアダプタでアレイディスクにアクセスする場合、そのうちの 2 セットが DMP92 として動作(2 重化)し、残る n-2 のディスクアダプタが DSP93 として動作する。

【0068】

94 は SVP27 に搭載する SVP 制御プログラムで、CHP91、DMP92、DSP93 を監視及び保守する

and it is something which electrical connection is done.

【0063】

As for 81, being mounted in downward part on substrate of host adapter 1, as for optical connector part, 82 which administers anti-upper position interface being mounted in the downward part on substrate of disk adapter 2, as for SCSI connector section, 83 which you connect with array disk 5 when mounting cache port package 22, it is a connector section for the connecting cable 23.

85 [kyashumemoripakkeji] is cache memory main body (cache memory 20 of Figure 2) which is installed in downward part of substrate (C) of 3.

【0064】

When \* removal doing each adapter, package with such as damage, operability in order to improve, it does not mount each connector section, to work surface 84 side excluding connector 83, [puratta] collection solid equipment has made the connecting side of 71.

【0065】

Figure 9 is figure which shows software constitution of this invention.

【0066】

91 is channel adapter control program which is written to boot device 12 of host adapter 1 (Below "CHP" with you call).

In addition, among disk adapter control program which are written to boot device 12 of the disk adapter 2, as for 92 disk adapter master control program which takes charge of data transfer control during treatment and cache memory and array disk of array disk peculiar (Below "DMP" with you call), 93 is disk adapter slave control program (Below "DSP" with you call) which takes charge of data transfer control between cache memory 20 and array disk 5 under control of DMP 92.

【0067】

2 kinds of DMP 92 and DSP 93 are written in boot device 12 of the disk adapter 2, but when with disk adapter of n set on equipment configuration access it makes array disk, 2 set among those operation (double conversion) it does as the DMP 92, disk adapter of n-2 which remains it operates as DSP 93.

【0068】

94 as with SVP control program which is installed in SVP27, CHP 91, DMP 92, DSP 93 is done supervision and

とともに、各制御プログラムの更新時は SVP27 から更新したい MP の制御プログラムを直接、または他の MP から当該 MP の制御プログラムを更新することができる。

[0069]

図 10 はデータの流れに基づいた図 9 で示したソフトウェア構成の機能分担を示した図である。

[0070]

CHP91 は、上位からのアドレス形式及びデータ形式を下位アドレス形式及びデータ形式に変換し、キャッシュメモリに書き込む。

101 はセグメント、102 はブロック、103 はアレイディスク 5 上の磁気ディスク 1 台当りに書き込むデータ量を表すストライプである。

DMP92 は、キャッシュメモリ上からストライプ単位にデータを読み出し、下位アドレスをアレイディスクの行 NO、列 NO、FBA、ブロック数に変換し、DSP93 でアレイディスクにデータを書き込む。

[0071]

また、DMP92 はアレイディスク 5 の構成情報も管理している。

[0072]

以上のように、各制御プログラムを機能分担することにより、上位インタフェースを SCSI やファイバーチャネル等に変更する場合は CHP91 のみ、またアレイディスク構成を変更(ディスクの行数/列数、RAID(Redundant Array Inexpensive Disk)方式等)する場合は DMP92 のみの変更で対応可能であり、ホストアダプタ 1、ディスクアダプタ 2 の接続変更に合わせて各制御プログラムを書き替えることで、スケールビリティを実現するとともに、ソフトウェア開発の負荷も軽減している。

[0073]

図 11 はコモンバス 4 の 2 重化の考え方と縮退動作を説明した図である。

[0074]

111 はコモンバス 4 の使用权を獲得することのできるバスマスタ(MP10 を搭載しているホストアダプタ 1 又はディスクアダプタ 2)、112 はバスマスタ 111 からのアクセス要求を受けるバススレーブ(キャッシュメモリパッケージ)である。

[0075]

conservation, when renewing each control program control program of MPa which you want to renew from SVP27 directly, or can renew control program of this said MPa from other MPa.

[0069]

Figure 10 is figure which shows function allocation of software constitution which is shown with Figure 9 which is based on flow of data.

[0070]

address form and data form from upper position it converts CHP 91, to the lower position address form and data form, writes to cache memory.

As for 101 as for segment, 102 as for block, 103 it is a stripe which displays data amount which is written to magnetic disk per machine on array disk 5.

DMP 92 from on cache memory data converts reading, lower position address to the quantity of line of array disk NO, line NO, FBA, block in stripe unit, to the array disk writes data with DSP 93.

[0071]

In addition, DMP 92 has managed also configuration information of array disk 5.

[0072]

Like above, when it modifies upper position interface in SCSI and fiber channel etc by function allocation doing each control program, only CHP 91, in addition when it modifies (number of lines / number of rows, RAID (Redundant array Inexpensive disk) system etc of disk) being correspondence possible with modification only of DMP 92, in combination with array disk configuration to modification of connection of host adapter 1, disk adapter 2, as by fact that each control program is rewritten, [sukeerabiriti] is actualized, You have lightened also load of software development.

[0073]

Figure 11 is way of thinking of double conversion of common bus 4 and figure which explains degeneracy operation.

[0074]

As for 111 bus master which can acquire use right of common bus 4 (host adapter 1 or disk adapter 2 which installs MPa 10), 112 is bus slave (cache memory package) which receives access request from bus master 111.

[0075]

Fバス26は通常動作状態では64ビットバス(200MB/S)2システムを同時に動作させ400MB/Sを実現しており、各バスシステムはパリティチェック又はタイムアウトで障害を検出可能である。

障害発生時はバスマスタ111は各自縮退状態に入り、残る1システムを使ってバススレーブをアクセスすると共に、この時の縮退情報は共有メモリ18上の管理エリアに登録される。

[0076]

またコモンバス内のシステム制御信号(バスリセット等)は信号線を3重化しており、通常動作時は3線一致、縮退動作時は2線一致(多数決)方式を採用することにより信頼性を高めている。

[0077]

図12は装置各部位における多重化と縮退運転を示した図である。

[0078]

121は2ポート化されたチャネルバスであり、ホストアダプタ1にはチャネルアダプタ29が2モジュール、対上位用のチャネルバスが4バス実装しており、障害発生時は交替チャネルアダプタ(CHP)、交替チャネルバスを使用して縮退運転に入る。

[0079]

122はディスクアダプタ2とアレイディスク5の間のインタフェースを司るSCSIバスで、1行の磁気ディスク群に対して別のディスクアダプタ2からもアクセス可能なように2重化しており、当バスに障害が発生した場合は交替SCSIバスを使用して縮退運転に入る。

また、アレイディスクマスタ制御を行うDMP92も2重化しており、障害発生時は交替DMP92を使用して縮退運転に入る。

[0080]

共有メモリ18、キャッシュメモリ20も2重化しており、共有メモリに障害が発生した場合は残るもう一方の使用して縮退運転に入り、キャッシュメモリに障害が発生した場合はライトペンディングデータ(キャッシュメモリ上に残っているデータ)をディスクにデステージし障害発生メモリ部位を除いたメモリで縮退運転を行う。

[0081]

アレイディスク5上の磁気ディスクに障害が発生した場合は、当該磁気ディスクを切り離し予備の磁気ディスクに修復しながら読み出し書き込

Fbus 26 usually with operating state 64 bit bus (200 MB /S ) 2 system operating simultaneously, has actualized 400 MB /S, each bus system damage is the inspectable with parity check or timeout .

At time of damage as for bus master 111 using 1 system which remains entering each degeneracy state , as access it does bus slave , as for degeneracy data at time of this it is registered to management area on joint ownership memory 18.

[0076]

In addition system control signal (bus reset etc) inside common bus to 3 heavy has converted the signal line , usual operation time at time of 3 line agreements and degeneracy operation raises reliability by adopting 2 line agreement (large number decision) system .

[0077]

As for Figure 12 it is a figure which shows multiplexing and degeneracy driving in device each site .

[0078]

To 2 port with channel pass which is converted, channel adapter 29 channel pass for 2 module , anti- upper position 4 pass has mounted 121 in host adapter 1, at time of damage replacement channel adapter (CHP) , using replacement channel pass , enters into degeneracy driving.

[0079]

With SCSI pass which administers interface between disk adapter 2 and the array disc 5, vis-a-vis magnetic disk group of one row even from another disk adapter 2 accessible way to double we convert 122, when damage occurs in this pass , using replacement SCSI pass , we enter into degeneracy driving.

In addition, to double we convert also DMP 92 which controls array disk master , at time of damage using replacement DMP 92, we enter into degeneracy driving.

[0080]

To double we convert also joint ownership memory 18, cache memory 20, when damage occurs in joint ownership memory , another which remains using, when damage occurs in cache memory entering degeneracy driving, [desuteeji ] wedesignate [raitopendingudeeta ] (data which remains on cache memory ) as disk and we drive degeneracy with memory which excludes damage memory section rank.

[0081]

Case damage occurs in magnetic disk on array disk 5, while separating the this said magnetic disk and rejuvenation designating as magnetic disk of preparatory it does the

み動作を行う。

【0082】

図 13 は装置の電源系の多重化と縮退運転を示した図である。

【0083】

商用電源制御部 58 は各々独立した AC 入力で 2 重化して、論理架用スイッチング電源 55 及びアレイディスク用スイッチング電源 57 にそれぞれ供給しているため、障害発生時はもう片方の商用電源制御部 58 で縮退運転に入る。

【0084】

131 は上位ホストからの電源 ON/OFF の遠隔制御や商用電源制御部 58、両スイッチング電源等の電源回路を制御する電源制御回路(以下「PCI 1」という)である。

【0085】

論理架用スイッチング電源 55 は冗長運転として必要数より 2 回路多く実装し電源コモンバスを介して論理架 51 及びバッテリー 52 に供給することにより、当スイッチング電源 55 が 2 回路故障しても動作可能である。

【0086】

同様に列単位の磁気ディスク群に供給するにアレイディスク用スイッチング電源 57 も、冗長運転として 2 回路多く実装し電源コモンバスを介して供給することにより、当スイッチング電源 57 が 2 回路故障しても動作可能であり、さらに両スイッチング電源 55、57 を 2 重化するよりも安価な構成に仕上げることができる。

【0087】

また停電時においては、2 重化されたバッテリー 52 から電源コモンバスを介して論理架内の揮発メモリであるキャッシュメモリ及び PCI 131 に供給され、片方のバッテリーが故障しても動作可能である。

【0088】

図 14 及び図 15 はアレイディスクに使用する磁気ディスク装置単体の記憶容量別にアレイディスクを構成したときのシステム性能を比較した図である。

【0089】

reading writing operation .

【0082】

Figure 13 is multiplexing of power supply system of device and figure which shows degeneracy driving.

【0083】

As for commercial power supply controller 58 each to double converting with AC input which becomes independent, because it has supplied to switching power supply 55 for logic rack and switching power supply 57 for array disk respectively, at time of damage already it enters into degeneracy driving with commercial power supply controller 58 of one side.

【0084】

131 is remote control of power supply ON/OFF from upper position host and power supply control circuit (Below "PCI " with you call ) which controls commercial power supply controller 58, both switching power supply or other power supply circuit .

【0085】

switching power supply 55 for logic rack twice road mounts more than necessary number as one for redundant driving and through power supply common bus , this switching power supply 55 twice road breaking down in logic rack by supplying 51 and the battery 52, it is a workable .

【0086】

It supplies to magnetic disk group of line unit in same way as one for redundant driving twice road mainly to mount also the switching power supply 57 for array disk , through power supply common bus , this switching power supply 57 twice road breaking down by supplying, with workable , finish \* \* it can designate thing as inexpensive configuration furthermore both switching power supply 55, 57 are converted the double with in comparison.

【0087】

In addition to double through power supply common bus from battery 52 which is converted, in time of electricity outage , it is supplied by cache memory and PCI 131 which are a volatilization memory inside logic rack, battery of one side breaks down and it is a workable .

【0088】

Figure 14 and Figure 15 when configuration doing array disk classified by recording capacity of magnetic disk device unit which is used for array disk , is figure which compares system performance .

【0089】

図 14 はそれぞれ異なる磁気ディスク装置を使用して同一容量のアレイディスクを実現した場合の構成を示しており、項番 141 が 3GB の磁気ディスク装置(3.5 インチ径のディスクを使用)、項番 142 が 4.0GB の磁気ディスク装置(5 インチ径のディスクを使用)、項番 143 が 8.4GB の磁気ディスク装置(6.4 インチ径のディスクを使用)を使用している。

アレイ構成は、ディスク装置 141 が 14 枚のデータディスクの 2 枚のパーティディスク、ディスク装置 142 が 14 枚のデータディスクと 4 枚のパーティディスク、ディスク装置 143 が 14 枚のデータディスクと 2 枚のパーティディスクで構成した場合である。

【0090】

図 15 は各磁気ディスク装置 141、142、143 についての毎秒当りの I/O 命令発行件数と平均応答時間の関係を示しており、アレイディスクシステムとしてのトランザクション性能を向上させるためには、小容量(小径)の磁気ディスク装置を使用してアレイ構成を大きくすることが最も性能を引き出せることから、本発明に於ては 3.5 インチ磁気ディスク装置 141 を採用してアレイディスクシステムを実現している。

従って、同じ記憶容量の磁気ディスク装置を、従来のように大形磁気ディスク装置 1 台で構成するのと、複数台の小形磁気ディスク装置のアレイで構成するのとでは、後者の小形磁気ディスク装置を多数用いたアレイ構成のものの方が、平均アクセスタイムを短縮できる点で有利である。

【0091】

以上説明してきたスケールラブルなアーキテクチャを使用して実現できる装置モデル構成例を図 16~図 19 にしめす。

【0092】

図 16 は、コモンバス 4 上のディスクアダプタ 2 の実装数を減らし、更にキャッシュポートパッケージ 22 を実装し、接続ケーブル 23 を介してキャッシュユニット 24 に接続することにより、キャッシュヒット率の高める高性能大容量キャッシュメモリ付小形ディスクアレイを実現した時の構成図である。

【0093】

またディスクアダプタ 2 を実装しないで、ホストアダプタ 1 とキャッシュメモリのみで構成した場合(図中の破線内の構成)は、記憶媒体が磁気ディスクから半導体メモリに代わり、更に高速データ

Figure 14 using different magnetic disk device respectively, has shown configuration when it actualizes array disk of same capacity, section turn 141 magnetic disk device of 3 GB (3.5 You use disk of inch diameter), section turn 142 magnetic disk device of 4.0 GB (You use disk of 5 inch diameter), section turn 143 has used magnetic disk device (6.4 You use disk of inch diameter) of 8.4 GB.

array configuration is, when disk drive 141 2 parity disk, disk drive 142 of 14 data disk 14 data disk and 4 parity disk, disk drive 143 configuration it does with 14 data disk and 2 parity disk.

【0090】

Figure 15 every concerning each magnetic disk device 141, 142, 143 has shown I/O command issue number of cases per second and relationship of even response time, in order transaction performance as array disk system to improve, using magnetic disk device of small capacity (small diameter), most it pulls out performance that it enlarges array configuration, \* from thing, Regarding to this invention, adopting 3.5 inch magnetic disk device 141, it actualizes array disk system.

Therefore, conventional way configuration it does magnetic disk device of same recording capacity, when with large scale magnetic disk device 1, configuration it does with array of the small shape magnetic disk device of plural table that with, those of array configuration which large number uses small shape magnetic disk device of the latter, is more profitable in point which can shorten even access thyme.

【0091】

Above using scaleable architecture which is explained, it shows device model configuration example which it can actualize in Figure 16 ~Figure 19.

【0092】

When actualizing high performance large capacity cache memory attaching small shape disk array which cache hit ratio raises by fact that Figure 16 decreases mount quantity of disk adapter 2 on the common bus 4, furthermore mounts cache port package 22, through connecting cable 23, connects to cache unit 24, it is a configuration diagram.

【0093】

In addition without mounting disk adapter 2, when configuration it does with only host adapter 1 and cache memory, (configuration inside dashed line of in the diagram) storage media furthermore actualizes high speed data transfer



転送可能な高性能の半導体ディスク装置を実現する。

【0094】

図 17 はディスクアダプタ 2 を最大構成とし、キャッシュパッケージ 3 を実装し又はキャッシュポート 22 を実装し接続ケーブル 23 を介してキャッシュユニットを接続することにより、高性能大容量キャッシュメモリ付大形ディスクアレイを実現した時の構成図である。

【0095】

図 18 はホストアダプタ 1 の対上位インターフェースを SCSI/ファイバーチャネル等のインターフェースに変えて、ディスクアダプタ 2 の実装数を減らし、更に Fバス 26 のビット幅を半分にした 2 系統で構成することにより、オープン市場をターゲットにした無停止運転の高性能フォールトレナント(高信頼性)サーバシステムを実現した時の構成図である。

【0096】

図 19 は図 18 の構成を元に 2 重化、活線挿抜を考慮せずに、最もシンプルな構成をとることによって安価なオープン市場向けのサーバシステムを実現した時の構成図である。

なお、図中、4D+1P は、データディスク 4 枚とパリティディスク 1 枚の趣旨である。

【0097】

以上の実施例において、コモンバス 4 上に、更に光ディスクアダプタ(光ディスク用接続論理モジュール)を介して光ディスク装置を接続し、磁気テープ制御装置(磁気ディスク接続論理モジュール)を介して磁気テープ装置を接続し、あるいは半導体記憶装置接続論理モジュールを介して半導体記憶装置を接続することができる。

また、コモンバス 4 上に別の形式のホストアダプタを介してワークステーションを接続することもできる。

このように、コモンバス上に、種々の形式の記憶装置に対する記憶媒体アダプタを接続することができる。

【0098】

【発明の効果】

以上詳しく説明したように、本発明によれば、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の

possible high performance semiconductor disk drive from magnetic disk in place of semiconductor memory .

【0094】

When actualizing high performance large capacity cache memory attaching large scale disk array by fact that Figure 17 designates disk adapter 2 as maximum configuration , mounts cache package 3 and or mounts the cache port 22 and through connecting cable 23, connects cache unit , it is a configuration diagram .

【0095】

When actualizing high performance [foorutoreranto ] (high reliability ) server system of non stop driving where the Figure 18 changing anti- upper position interface of host adapter 1 into SCSI /fiber channel or other interface , decreasesmount quantity of disk adapter 2, designates open market as target furthermore by configuration doing with 2 system which reduce bit width of Fbus 26 in half , it is a configuration diagram .

【0096】

As for Figure 19 when on basis of configuration of Figure 18 withoutconsidering double conversion and live wire \* removal, mostactualizing server system for inexpensive open market by fact that simple configuration istaken, it is a configuration diagram .

Furthermore, in the diagram , 4D+1P data disk 4 and is gist of parity disk one layer .

【0097】

In Working Example above, on common bus 4, furthermore through optical disk adapter (Connected logic module for optical disk ),optical disk device can be connected, through magnetic tape controller (magnetic disk connected logic module ), magnetic tape device can beconnected, or through semiconductor storage device connected logic module , semiconductor storage device can beconnected.

In addition, through host adapter of another form on common bus 4, it canalso connect workstation .

this way, on common bus , storage media adapter for storage device of various form canbe connected.

【0098】

[Effects of the Invention ]

As above explained in detail, according to this invention , in storage system which possesses cache memory device (upper side connected logic device of plural and cache memory device which is sharedto storage device side connected logic

の記憶装置側接続論理装置と、これらの装置間で転送されるデータを一時記憶するキャッシュメモリ装置(複数の上位側接続論理装置及び複数の記憶装置側接続論理装置に共有されるキャッシュメモリ装置)とを有する記憶システムにおいて、前記複数の上位装置側接続論理装置、複数の記憶装置側接続論理装置、及びキャッシュメモリ装置は、これらの装置に共有される共通バスにより相互に接続されるように構成したので、上位側接続論理装置と記憶装置側接続論理装置とキャッシュメモリの増設または変更は、単に共通バス上にこれらの装置等を追加しまたは変更して行くだけでよく、増設によるアップグレードが容易に達成できスケラブルなシステム構成を得ることができる。

また、これらの上位側接続論理装置、記憶装置側接続論理装置及びキャッシュメモリ装置は、モジュール化されて、共通バスの配設されたプラッタに挿抜(着脱)自在に取り付けるようにしたので、これらの装置の必要な数量の増設作業も簡単であるという効果がある。

【0099】

また、上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置、及びこれらの間を接続する共通バスは、二重化され、2系統に分けて配線されているので、これらの装置の一方に障害が発生したときでも、他方の装置を用いて縮退運転が可能である。

この場合、上位側接続論理装置、記憶装置側接続論理装置、及びキャッシュメモリ装置は、いずれも活線挿抜対応のコネクタ部を具備しているため、システムを停止することなく保守点検を行なって故障部品の交換を行ったり、増設用の部品を追加したりすることが可能であるという効果がある。

【0100】

更に、記憶装置は、複数の小形記憶装置を組み合わせたアレイ形とされ、これにより従来の大形ディスク装置 1 台を用いたものに比べてアクセスタイムを短縮できるという効果がある。

【0101】

また、キャッシュメモリ装置は、共通バスに直接取り付けられるキャッシュメモリモジュール(キャッシュメモリパッケージ)と、増設用のキャッシュユニットとで構成され、増設用のキャッシュユニットは、共通バスに直接挿抜自在に取り付け

device of plural ) which remembers storage device side connected logic device of plural which configuration does upper side connected logic device of plural which configuration does interface for upper position device and interface for storage device and aforementioned storage device and data which is transferred between device of these at one time, Because in order to be connected mutually, by common bus which is shared to these device configuration it did storage device side connected logic device , and cache memory device of upper position equipment side connected logic device , plural of aforementioned plural , as for upper side connected logic device and storage device side connected logic device and addition or modification of cache memory , Simply, it can add these device etc on common bus and or it just modifies to be necessary, can achieve up grade easily with addition and can acquire scaleable system configuration .

In addition, modulization being done, \* removal (attachment and detachment ) you install these upper side connected logic device , storage device side connected logic device and cache memory device , unrestrictedly in [puratta ] where common bus is arranged, because it required, there is an effect that also addition job of required number quantity of these device is simple.

【0099】

In addition, upper side connected logic device , storage device side connected logic device , cache memory device , and the common bus which connects between these to be done doubling , dividing into 2 system , because wiring it is done, when these device damage occurs on one hand even, degeneracy driving is possible making use of the device of other .

In case of this , because upper side connected logic device , storage device side connected logic device , and cache memory device in each case possess connector section of live wire \*removal correspondence, doing preservation inspection without stopping system ,there is an effect that it is possible to exchange breakdown part , to add part for addition.

【0100】

Furthermore, there is an effect that storage device makes array shape which combines small shape storage device of plural , can shorten access time in comparison with those which because of this use conventional large scale disk drive 1 .

【0101】

In addition, as for cache memory device , cache memory module which is directly installed in common bus (cache memory package ) with, configuration to be done with cache unit for addition, as for cache unit for addition, through cache port package for the addition which, \* removal is installed

られる増設用のキャッシュポートパッケージを介して必要数接続されるようになっているので、簡単に増減することができるという効果も得られる。

【0102】

以上により、高信頼性の記憶システムを得ることができる。

【図面の簡単な説明】

【図1】

本発明の実施例の概要を示す概念図である。

【図2】

本発明の一実施例の記憶システムの詳細な構成図である。

【図3】

図2の構成図に沿ったデータの流れとデータ形式を示した図である。

【図4】

本発明の一実施例の装置外観図である。

【図5】

本発明の一実施例の装置における制御ユニット部の実装方式図である。

【図6】

本発明の一実施例の装置におけるアレイディスクユニット部の実装方式図である。

【図7】

本発明の一実施例の装置における論理架部の接続方式図である。

【図8】

本発明の一実施例の装置における論理架部の実装方式図である。

【図9】

本発明の実施例に適用されるソフトウェア構成図である。

【図10】

本発明の実施例によるデータの流れとソフトウェアの機能分担を示した図である。

【図11】

本発明の実施例によるコンバスの2重化と縮

directly unrestrictedly in the common bus necessary number it is connected, because it groans, also theeffect that is acquired it can increase and decrease simply.

[0102]

With above, storage system of high reliability can be acquired.

[Brief Explanation of the Drawing (s)]

[Figure 1 ]

It is a conceptual diagram which shows gist of Working Example of this invention .

[Figure 2 ]

It is a detailed constitution figure of storage system of one Working Example of this invention .

[Figure 3 ]

It is a figure which shows flow and data form of data whichparallels to configuration diagram of Figure 2 .

[Figure 4 ]

It is a device external view of one Working Example of this invention .

[Figure 5 ]

It is a mounting system figure of control unit section in device of one Working Example of this invention .

[Figure 6 ]

It is a mounting system figure of array disc unit section in device of one Working Example of this invention .

[Figure 7 ]

It is a connection system figure of logic rack section in device of the one Working Example of this invention .

[Figure 8 ]

It is a mounting system figure of logic rack section in device of the one Working Example of this invention .

[Figure 9 ]

It is a software constitution figure which is applied to Working Example of this invention .

[Figure 10 ]

It is a figure which shows function allocation of flow and software of the data with Working Example of this invention .

[Figure 11 ]

It is a figure which shows double conversion and degeneracy

退動作を示した図である。

【図12】

本発明の実施例による装置各部位の 2 重化と縮退運転を示した図である。

【図13】

本発明の実施例による装置の電源系の多重化と縮退運転を示した図である。

【図14】

アレイディスクに使用する磁気ディスク装置単体のディスク構成を示す図である。

【図15】

磁気ディスク装置の記憶容量とアレイディスクのシステム性能を示した図である。

【図16】

高性能大容量キャッシュメモリ付小形ディスクアレイの構成図である。

【図17】

高性能大容量キャッシュメモリ付大形ディスクアレイの構成図である。

【図18】

高性能フォールトトレラントサーバシステムの構成図である。

【図19】

低価格サーバシステムの構成図である。

【図20】

従来の記憶システムの概略構成図である。

### Drawings

【図1】

operation of common bus with Working Example of this invention .

[Figure 12 ]

It is a figure which shows double conversion and degeneracy driving of device each site with Working Example of this invention .

[Figure 13 ]

It is a multiplexing of power supply system of device and a figure which shows degeneracy driving with Working Example of this invention .

[Figure 14 ]

It is a figure which shows disk configuration of magnetic disk device unit which is used for array disk .

[Figure 15 ]

It is a recording capacity of magnetic disk device and a figure which shows system performance of the array disk .

[Figure 16 ]

It is a configuration diagram of high performance large capacity cache memory attaching small shape disk array .

[Figure 17 ]

It is a configuration diagram of high performance large capacity cache memory attaching large scale disk array .

[Figure 18 ]

It is a configuration diagram of high performance [foorutotoreantosaabashisutemu ] .

[Figure 19 ]

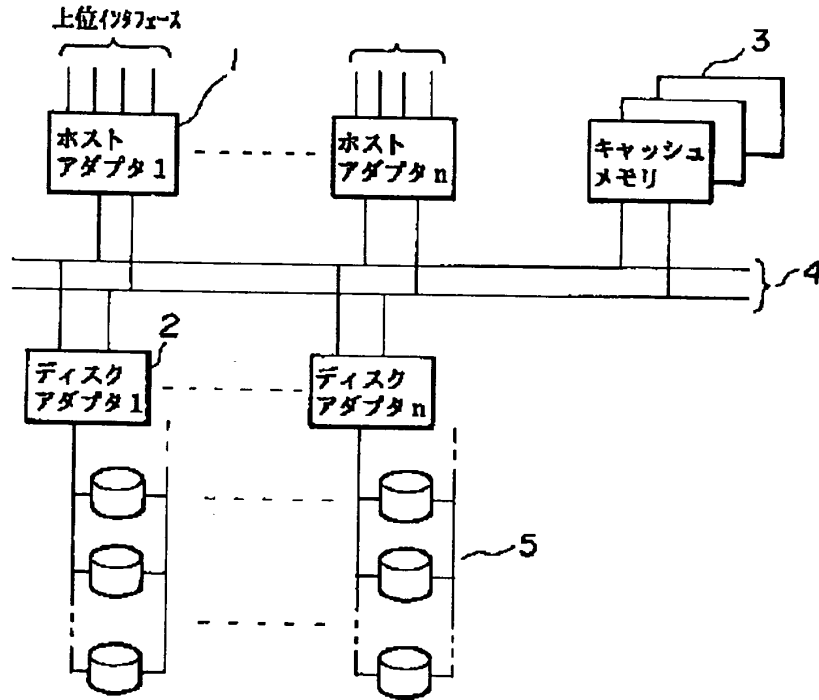
It is a configuration diagram of low cost server system .

[Figure 20 ]

It is a conceptual constitution diagram of conventional storage system .

[Figure 1 ]

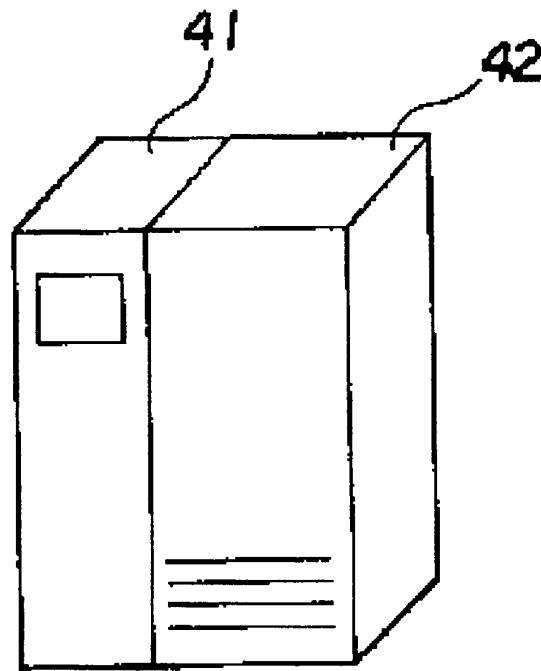
【図 1】



【図4】

[Figure 4]

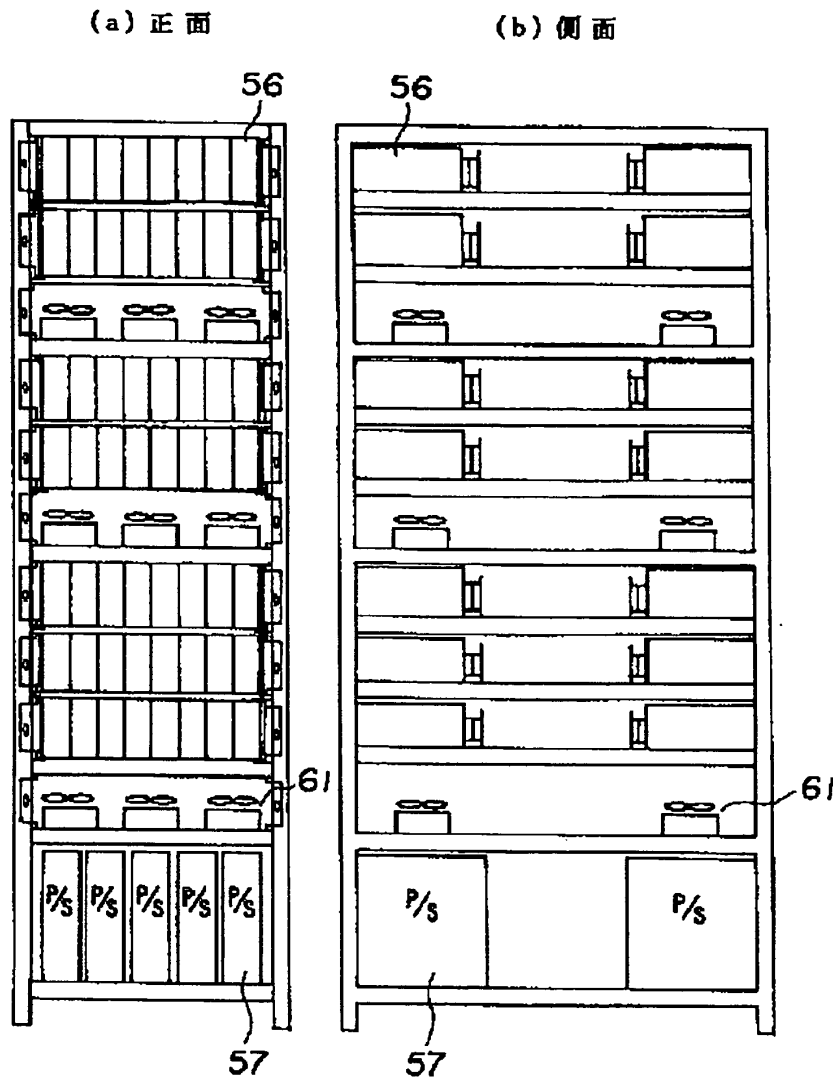
【 図 4 】



【 図 6 】

[Figure 6]

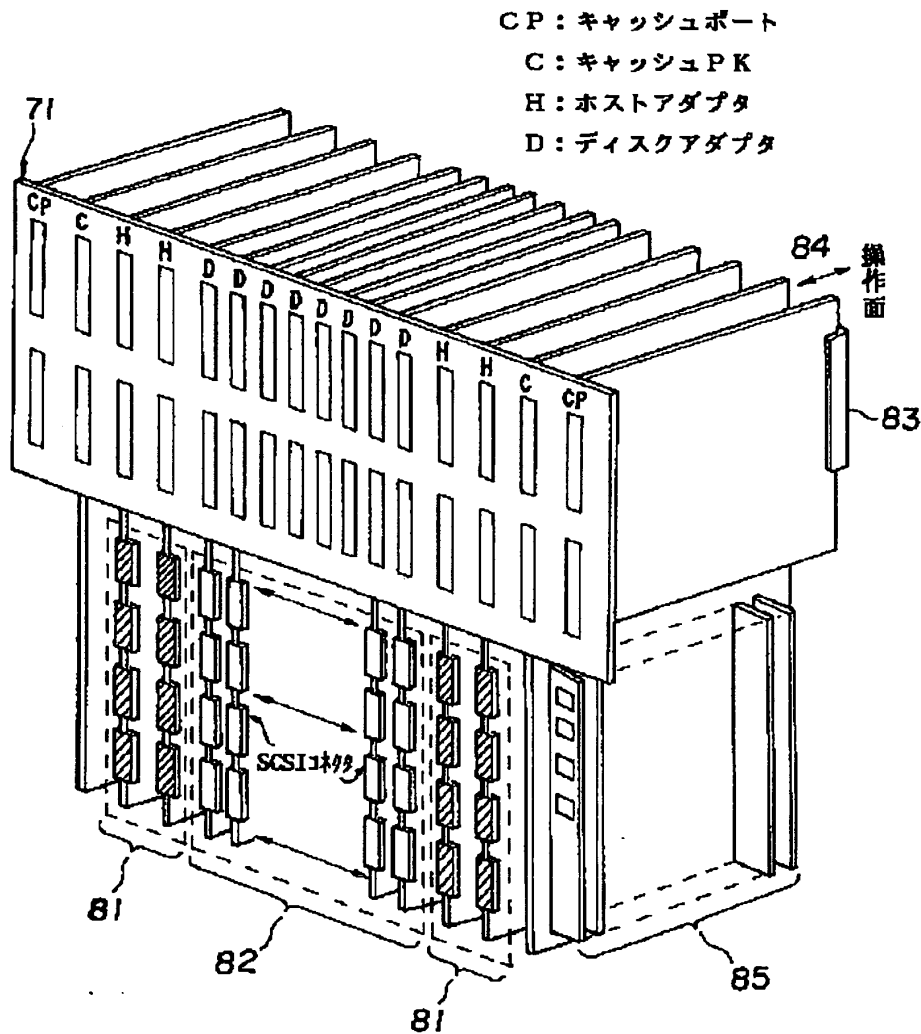
【図6】



【図8】

[Figure 8]

【図8】

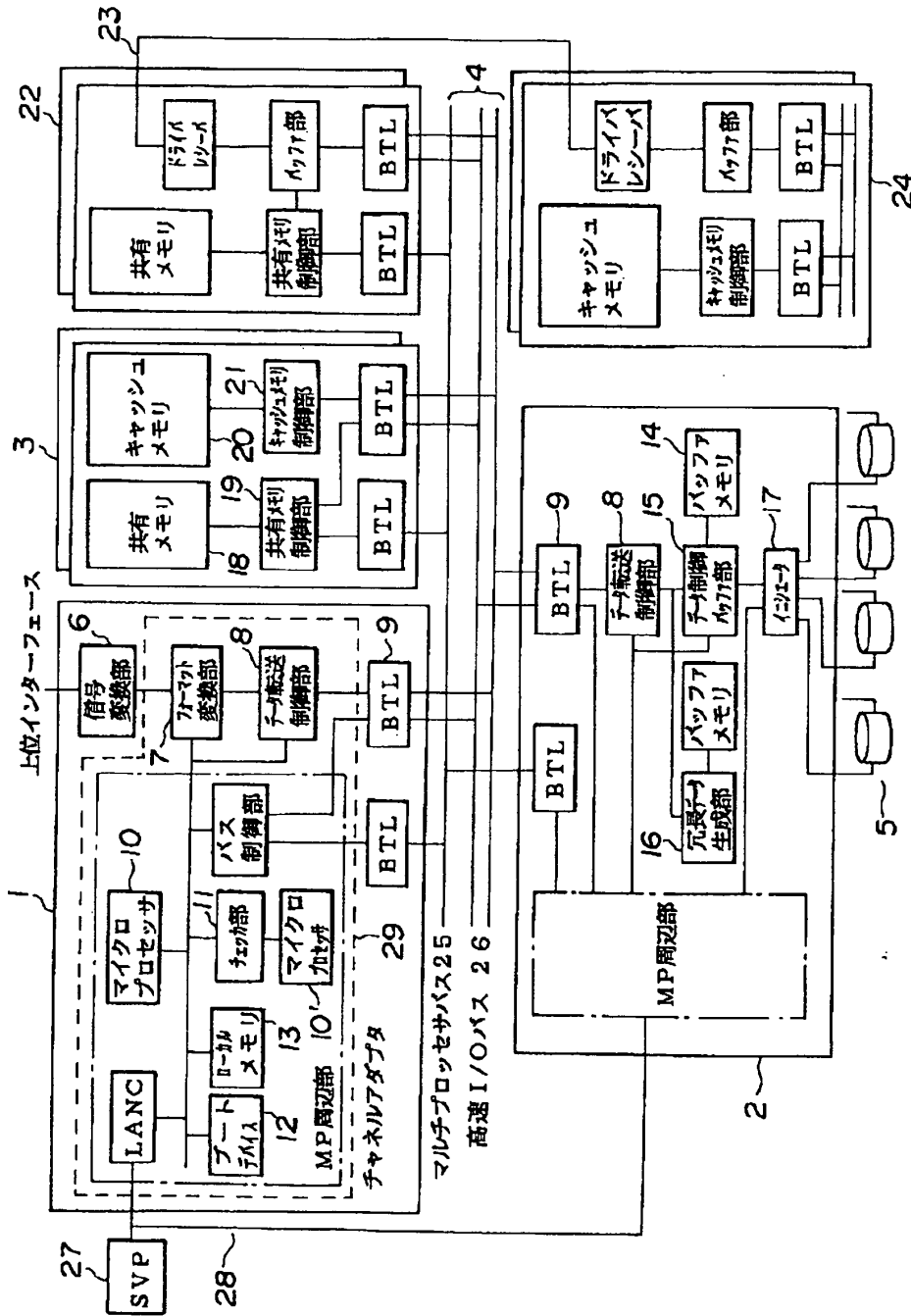


【図2】

[Figure 2]



【図2】



(7,296)

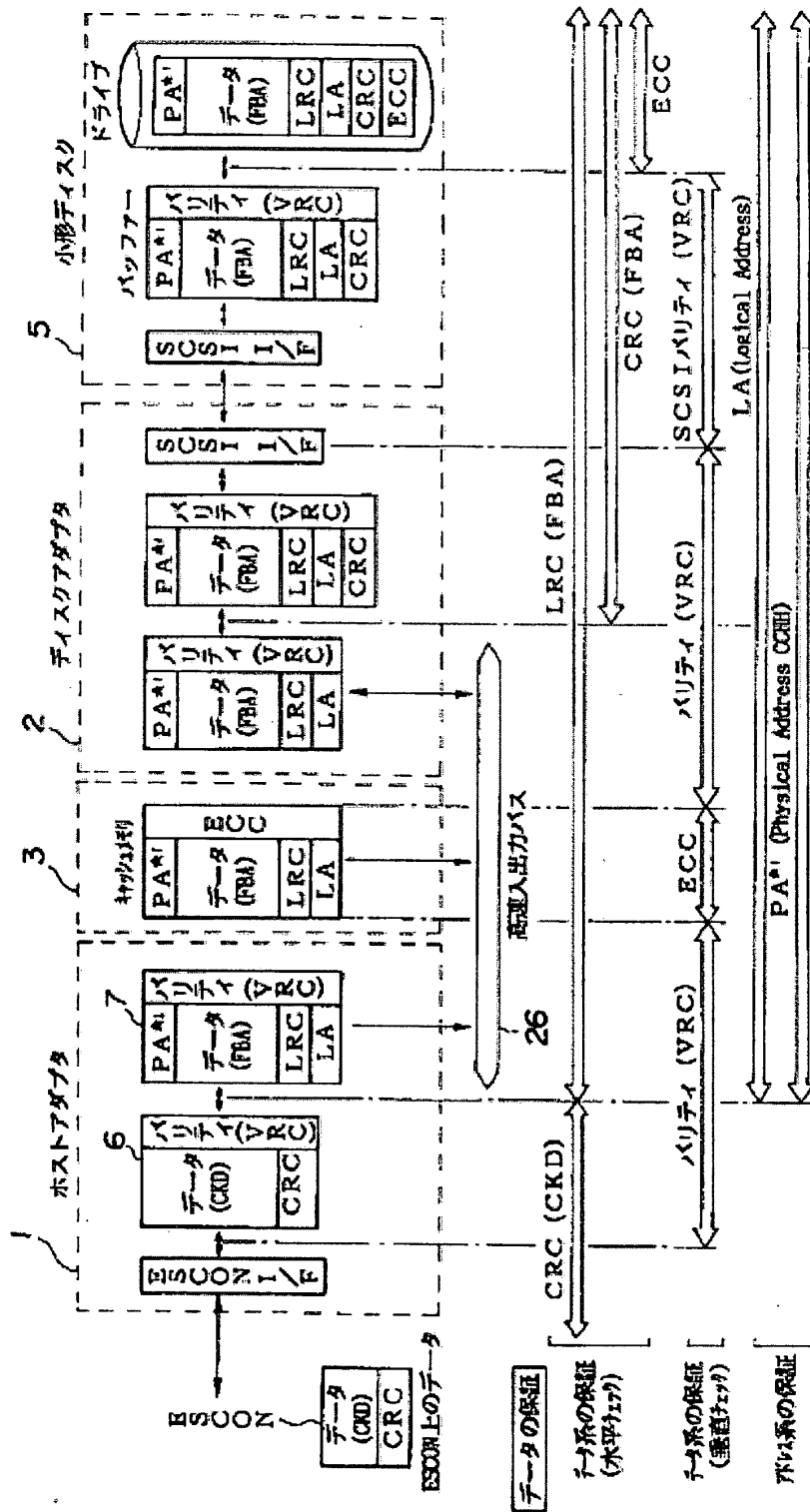
**JP1995020994A**

**1995-1-24**

**【図3】**

**[Figure 3 ]**

【図3】

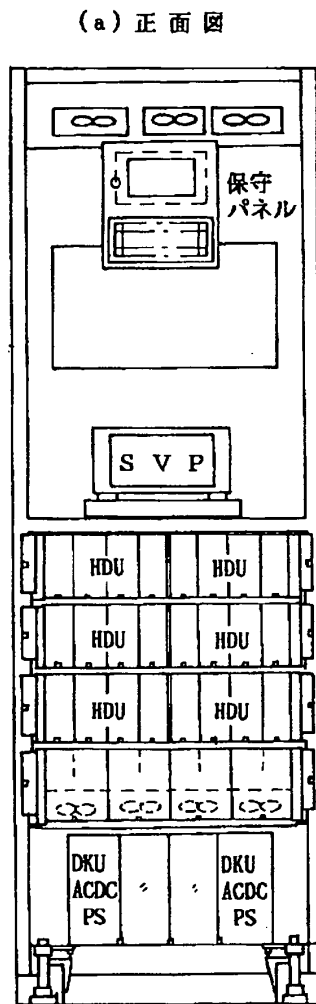


(\*) PAはCKDDのカウント値に対応するブロック上のみ存在。

【図5】

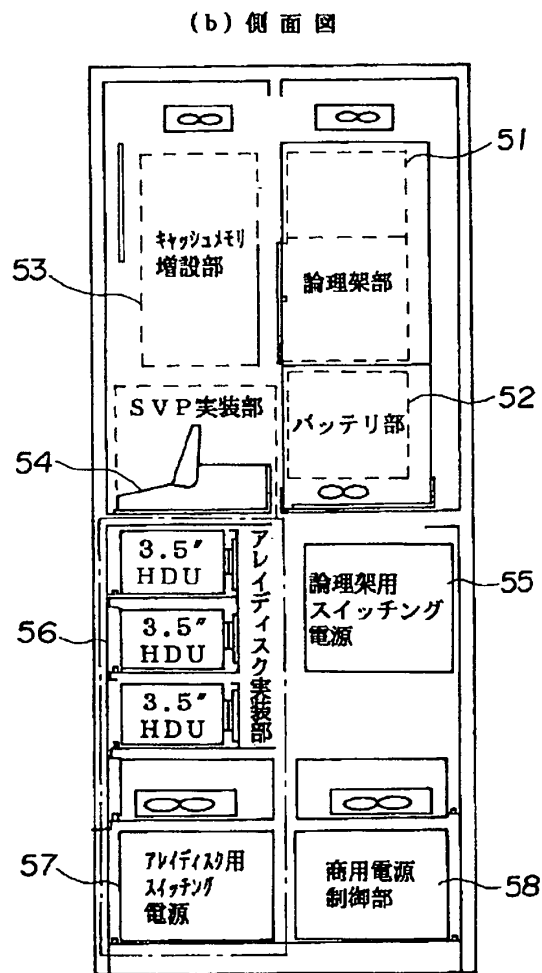
[Figure 5]

【図5】



【図14】

[Figure 14]



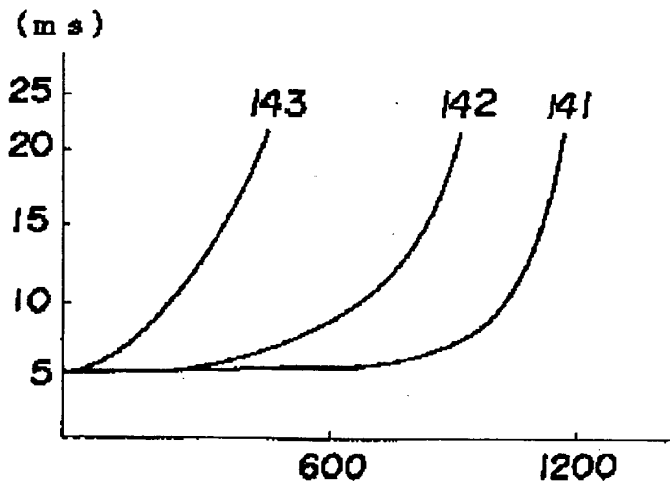
【図14】

項番	磁気ディスク単体容量	アレイ構成	アレイ容量
141	3.0GB (3.5インチ)	(14D+2) × 5	約220GB
142	4.0GB (5インチ)	(14D+2) × 4	
143	8.4GB (6.4インチ)	(14D+2) × 2	

【図15】

[Figure 15]

【図15】

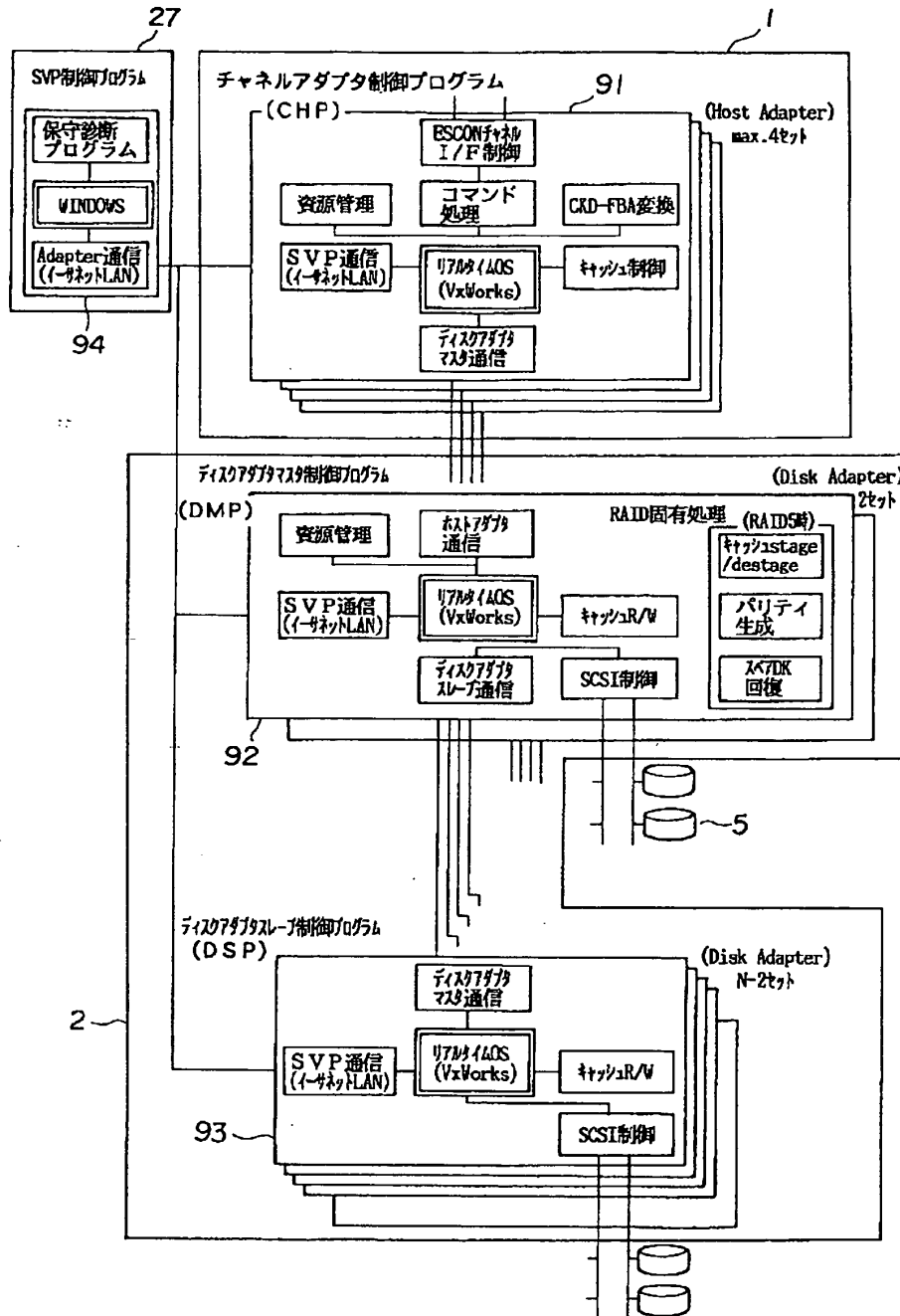


【図7】

[Figure 7]



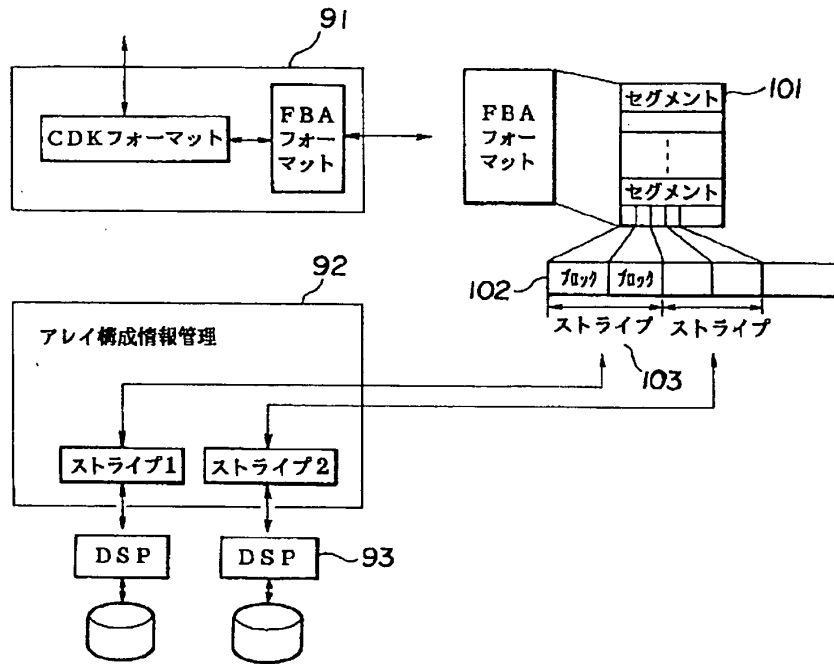
【図9】



Page 2 of 2  
Copyright © 1995 by International Business Machines Corporation. U.S. Pat. No. 5,470,276, Pat. Pending Ser. No. 08/507,296

【図10】

[Figure 10]



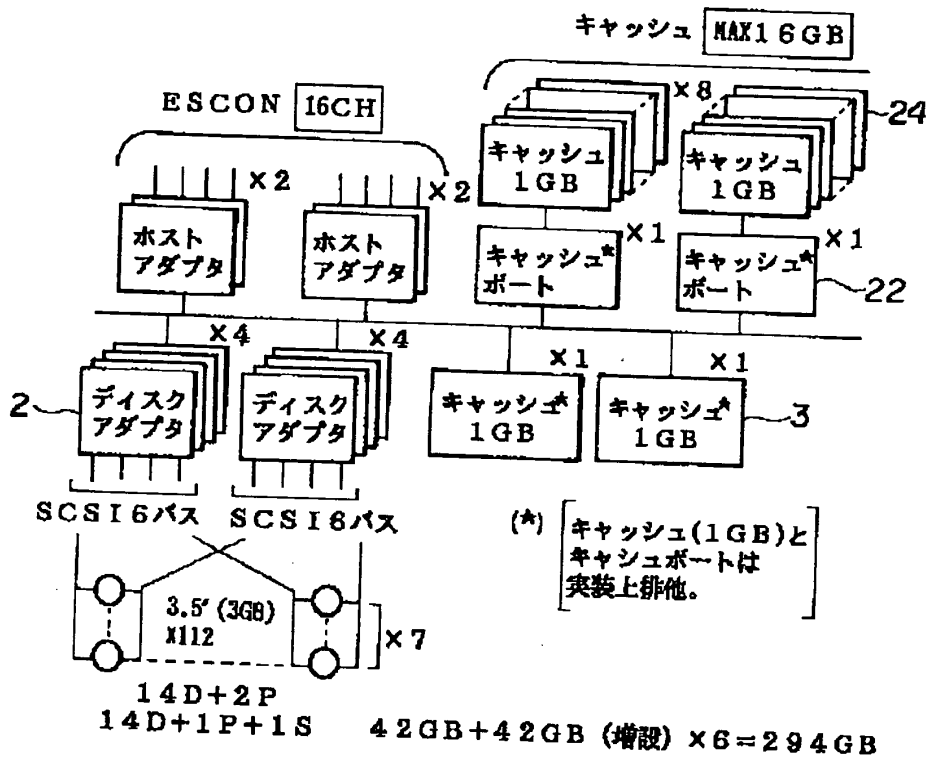
【図10】

【図17】

[Figure 17]



【図17】

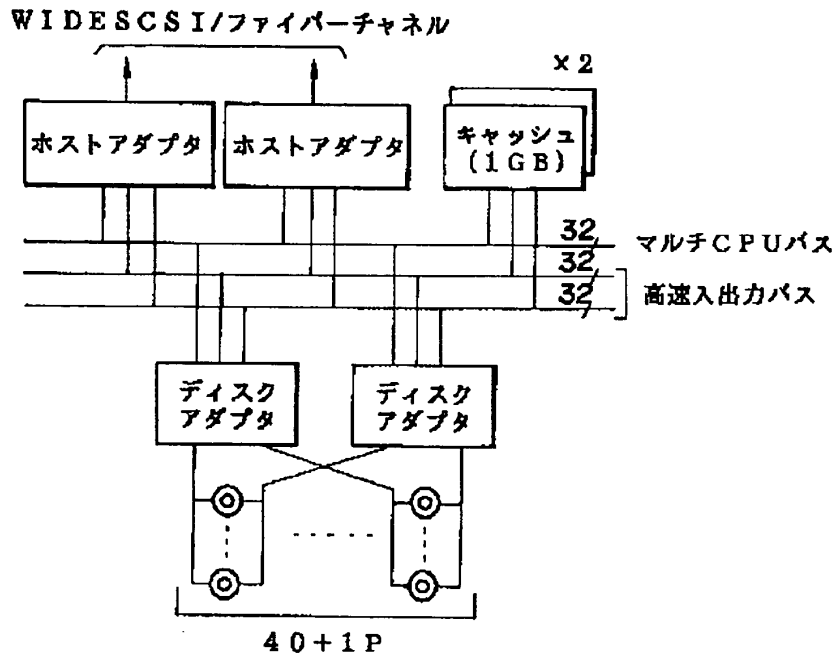


【図18】

[Figure 18]

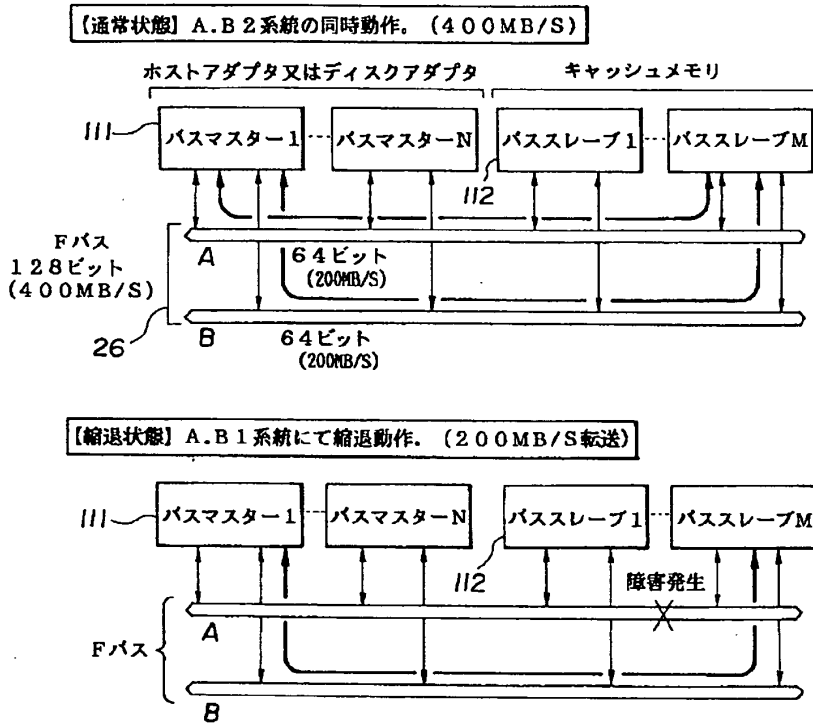
【図18】

- RAID5/3/1/0
- システムキャッシュサポート
- フォールトトレラント



【図11】

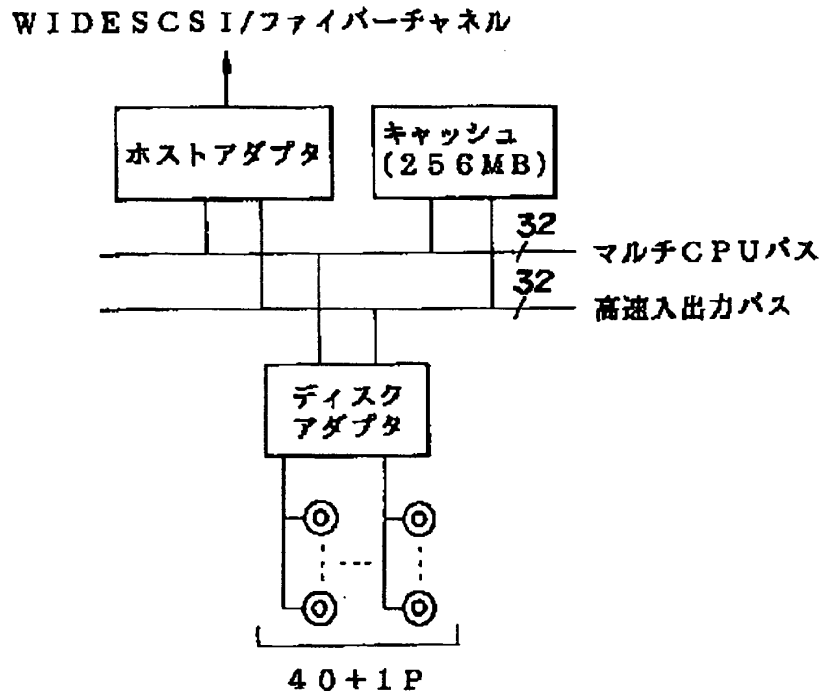
[Figure 11]



【図19】

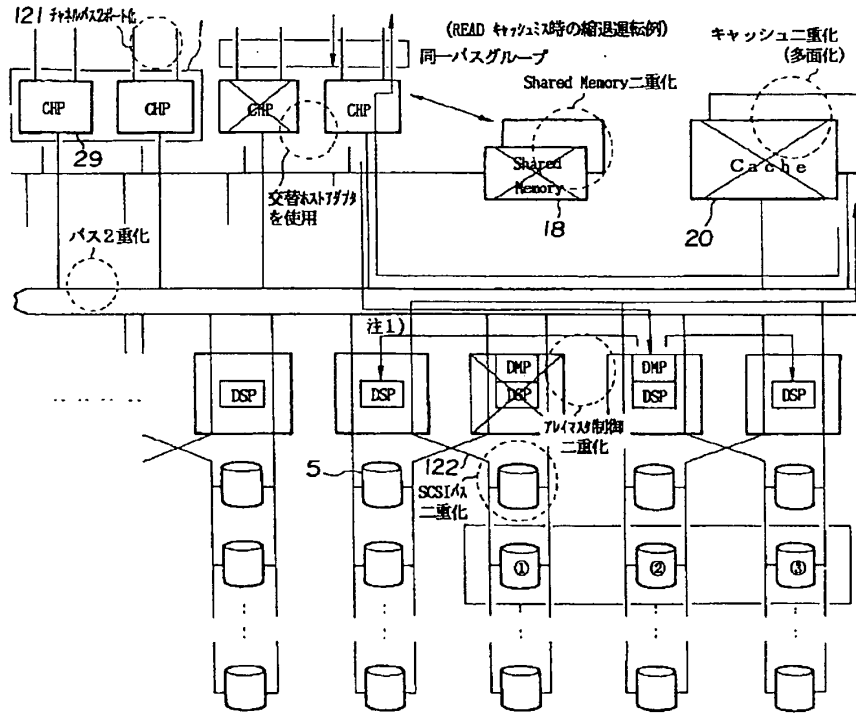
[Figure 19]

【図19】



【図12】

[Figure 12]

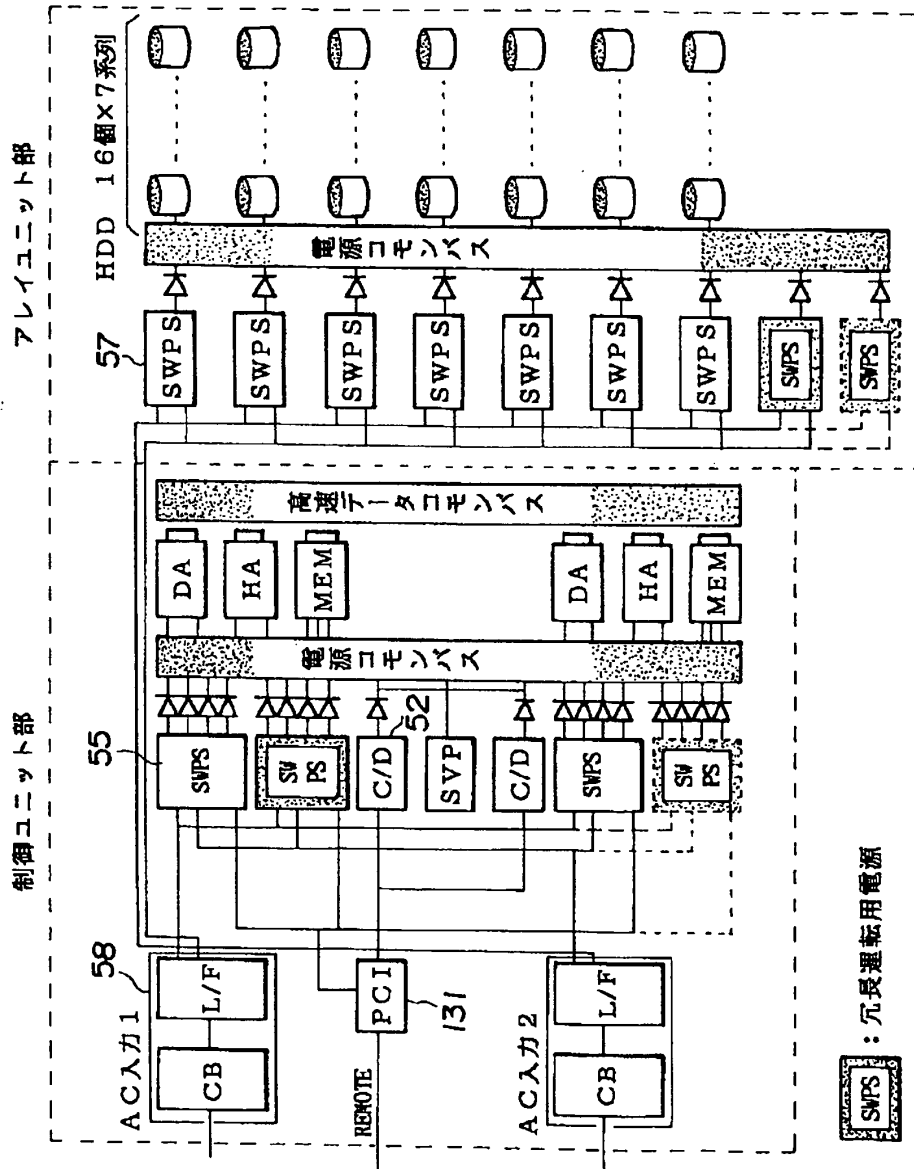


【図12】

【図13】

[Figure 13]

【図13】

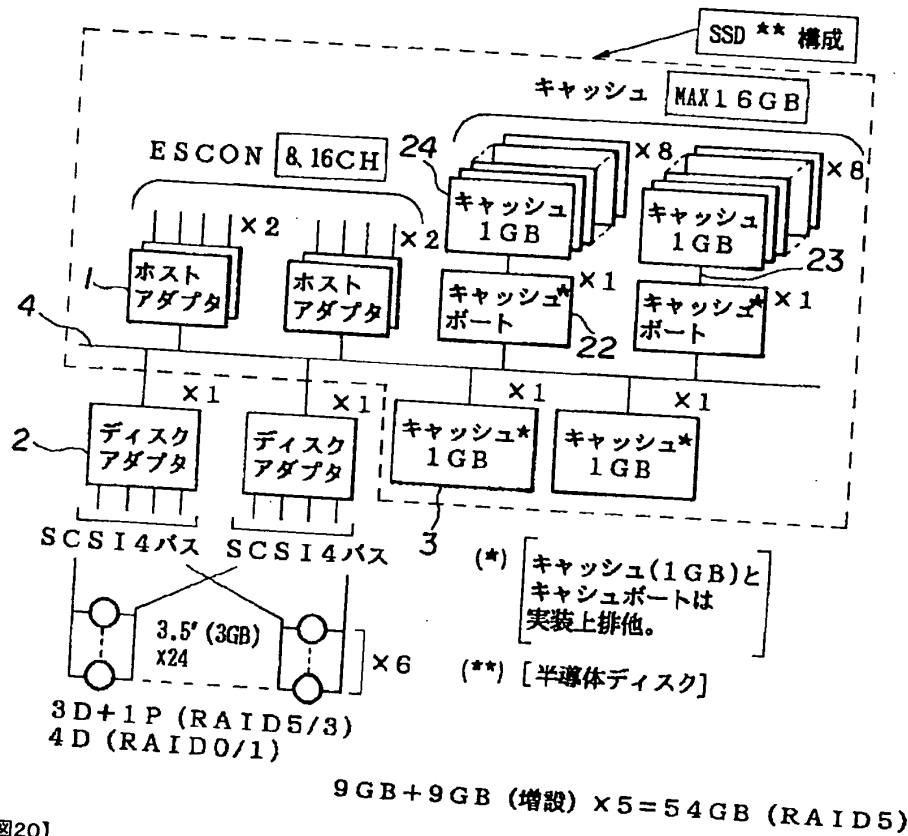


JP1995020994A

【図16】  
【図16】

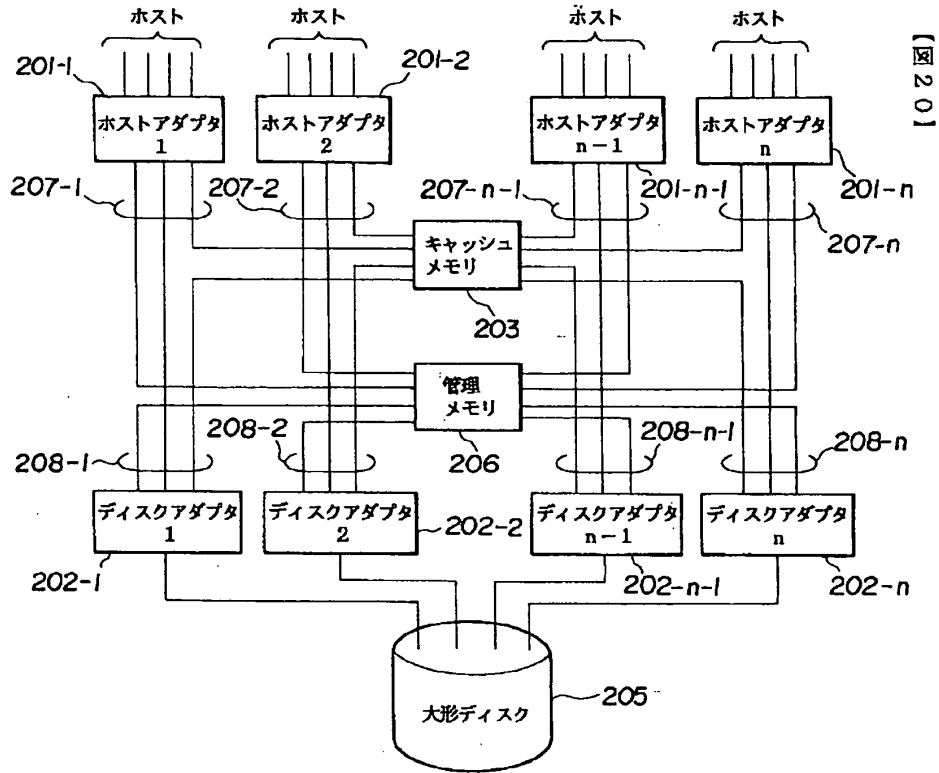
[Figure 16]

1995-1-24



【図20】

[Figure 20]





(19)



JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 07020994 A

(43) Date of publication of application: 24.01.95

(51) Int. Cl. G06F 3/06  
G06F 12/08  
G06F 13/12

(21) Application number: 05162021

(22) Date of filing: 30.06.93

(71) Applicant: HITACHI LTD

(72) Inventor: NINOMIYA TATSUYA  
MASUZAKI HIDEFUMI  
KUROSAWA HIROYUKI  
TAKAHASHI NAOYA  
INOUE YASUO  
IWASAKI HIDEHIKO  
HOSHINO MASAYUKI  
ISONO SOICHI

(54) STORAGE SYSTEM

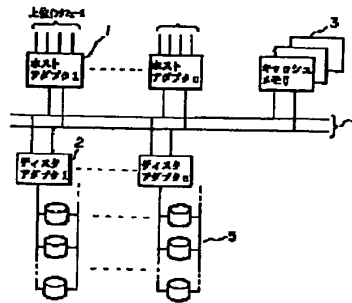
(57) Abstract

PURPOSE: To easily extend and vary the scale of the storage system of a large-sized computer and to maintain the system by degeneration and hot-line insertion and extraction.

CONSTITUTION: Plural host adapters (host-side interface) 1 connected to a host CPU, plural disk adapters (storage device side interface) 2 connected to an array disk 5, and cache memories 3 for temporary storage that are shared by those adapters are fitted onto common buses 4 that are shaved by those adapters and cache memories in a free insertion and extraction state. For an extension of the scale, adapters 1 and 2 and cache memories 3 which are as many as required are only added. The adapters 1 and 2, cache memories, and common buses are duplexed to enable degenerative operation in the case of a fault, and the joint parts between the adapters and cache memories, and common buses are so constituted that hot-line insertion and extraction are enabled and

maintenance inspection and component replacement can be done without stopping the system.

COPYRIGHT: (C)1995,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-185594

(43) 公開日 平成9年(1997)7月15日

(51) Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 15/163			G 0 6 F 15/16	3 2 0 R 3 2 0 V

審査請求 未請求 請求項の数6 OL (全 25 頁)

(21) 出願番号	特願平8-301770	(71) 出願人	391058071 タンデム コンピューターズ インコーポ レイテッド TANDEM COMPUTERS IN CORPORATED アメリカ合衆国 カリフォルニア州 95014 クーパーティノ ノース タンタ ウ アベニュー 10435
(22) 出願日	平成8年(1996)11月13日	(72) 発明者	トッド ダブリュー スプレングル アメリカ合衆国 カリフォルニア州 94086 サニーヴェイル ヴァスケズ ア ベニュー 387
(31) 優先権主張番号	08/556618	(74) 代理人	弁理士 中村 稔 (外6名)
(32) 優先日	1995年11月13日		
(33) 優先権主張国	米国 (US)		

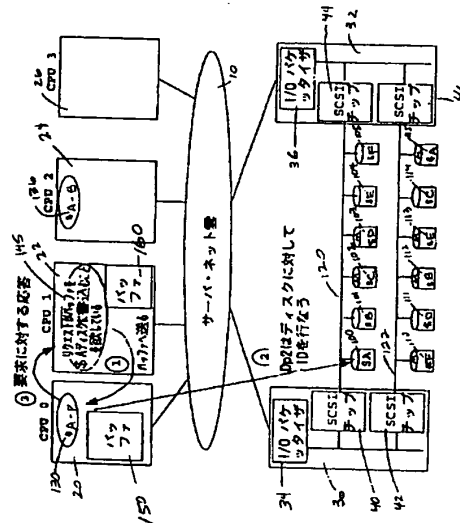
最終頁に続く

(54) 【発明の名称】 直接バルク・データ転送

(57) 【要約】 (修正有)

【課題】 CPUsと記憶ディスクとの間のデータの転送中の類似するデータ・コピーを除去する。

【解決手段】 記憶処理130は、記憶装置100~105及び110~115を有する30へのアクセスを制御する。ソフトウェア・ルーチンは、要求CPU22による記憶装置30への直接アクセスを供給するために用いられる。要求CPU22のバッファ160に対する仮想メモリ・アドレスは、要求CPU22で生成される。記憶装置アクセス要求と共に仮想メモリ・アドレスは、記憶処理130を含んでいるCPU20に送られる。ワーク要求は、記憶処理130から記憶装置30へ送られる仮想メモリ・アドレスを含む。次いで、データは、要求CPU22と記憶装置30との間で直接転送される。記憶装置30は、次いでワーク要求に応答する。



【特許請求の範囲】

【請求項1】 上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置及び前記複数の記憶装置側接続論理装置間で転送されるデータを一時記憶するキャッシュメモリ装置とを有する記憶システムにおいて、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、これらの装置に共用されるコモンバスにより相互に接続されるように構成したことを特徴とする記憶システム。

【請求項2】 前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、いずれもモジュールで構成し、前記モジュールは、それぞれ、前記コモンバスに対し挿抜自在に取付けられるように構成したことを特徴とする請求項1記載の記憶システム。

【請求項3】 前記コモンバスは、ブラッタ上に配設され、前記上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記ブラッタに対し挿抜自在に取付けられるように構成したことを特徴とする請求項1または2記載の記憶システム。

【請求項4】 前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記コモンバスは、いずれも少なくとも二重化されており、前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記コモンバスの一方により縮退運転が可能となるように構成したことを特徴とする請求項1ないし3のいずれか1記載の記憶システム。

【請求項5】 前記二重化された上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置は、いずれも活線挿抜ができるように構成したことを特徴とする請求項4記載の記憶システム。

【請求項6】 前記記憶装置は、二重化された電源部を備えたことを特徴とする請求項1ないし6のいずれか1記載の記憶システム。

【請求項7】 前記記憶装置は、複数の小形記憶装置を組み合わせたアレイ記憶装置で構成したことを特徴とする請求項1ないし3のいずれか1記載の記憶システム。

【請求項8】 前記キャッシュメモリ装置は、キャッシュメモリ本体を持ち、前記コモンバスに直接取り付けられるキャッシュメモリモジュールと、キャッシュメモリを持つ増設用のキャッシュユニットとを有しており、前記キャッシュユニットは、前記コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して接続されるように構成したことを特徴とする請求項1～7のいずれか1記載の記憶システム。

【請求項9】 前記上位側接続論理装置及び前記記憶装置側接続論理装置は、それぞれ、二重化されたマイクロプロセッサを有し、両マイクロプロセッサによりデータの比較チェックを行なうように構成したことを特徴とする請求項1ないし8のいずれか1記載の記憶システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、大形計算機システムやネットワークシステム等に接続される磁気ディスク装置、磁気テープ装置、半導体記憶装置、または光ディスク装置等の記憶装置を制御する記憶制御装置を含む記憶システムに係り、特に、システムの拡張性が高く縮退運転や活線挿抜対応の可能な記憶システムに関する。

【0002】

【従来の技術】従来、大形計算機に接続される記憶システムとして、例えば特開昭61-43742号公報に記載されているように、上位装置(CPU)に対するインタフェース(ホストアダプタ)、キャッシュメモリ、及び磁気ディスク装置等の記憶装置に対するインタフェース(ディスクアダプタ)の相互間をホットライン(専用線)で接続しているものが知られている。

【0003】図20は、従来の記憶システムの構成の概要を示す図である。同図において、201-1~201-nはそれぞれ複数の上位ホスト(CPU)に接続されるホストアダプタ(対上位論理モジュール)、202-1~202-nは、共有の大形ディスク装置205に接続されるディスクアダプタ(記憶媒体接続用論理モジュール)、203は、複数のホストアダプタに共有のキャッシュメモリ、206は同様に共有の管理メモリである。従来装置では、各ホストアダプタ201-1~201-nとキャッシュメモリ203の間、キャッシュメモリ203と各ディスクアダプタ202-1~202-nの間、各ホストアダプタ201-1~201-nと管理メモリ206の間、並びに管理メモリ206と各ディスクアダプタ201-2~201-nの間は、それぞれ別々のホットライン207-1~207-n及び208-1~208-nによって接続されている。また、これらのホストアダプタ及びディスクアダプタの監視及び保守を行なう保守用プロセッサ(SVP、図示せず)も各々のホストアダプタ及びディスクアダプタにそれぞれ専用線を介して接続されている。

【0004】

【発明が解決しようとする課題】上記従来技術では、上位装置に対するホストアダプタ(対上位接続論理モジュール)と、記憶装置に対するディスクアダプタ(対記憶媒体接続論理モジュール)と、キャッシュメモリ(キャッシュメモリモジュール)との各間がホットラインで接続されているため、装置構成が複雑になると共に、ホストアダプタ、キャッシュメモリ、ディスクアダプタ、ディスク装置等、装置としての拡張性に乏しくいわゆるス

ケーラブル(拡張及び縮小自在)なシステム構成が得られなかった。また、システムを多重化することにより障害発生時等に縮退運転(2台のうち1台を停止し他の1台だけで運転するなど)や活線挿抜対応(システムを動作したままで基板や回路の部品等を挿しかえるなど)を可能とすることがなにも配慮されておらず、このため、障害発生時の部品交換やシステムの制御プログラムをグレードアップするときには、システムを一時停止し対応しなければならない等の問題があった。

【0005】従って、本発明の目的は、上記従来技術の問題点を解決し、コモンバス方式を採用することにより、システム構成(規模)に応じてホストアダプタ、記憶装置アダプタ等の各論理モジュールやキャッシュメモリ及び記憶媒体を接続することでスケラブルなシステムを実現することができるようにすると共に、各論理モジュール、記憶媒体及びコモンバスの多重化により、縮退運転と各論理モジュール及び記憶媒体の活線挿抜対応とを可能とし、無停止で保守することができる記憶システムを提供することにある。

【0006】

【課題を解決するための手段】上記目的を達成するため、本発明は、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置及び前記複数の記憶装置側接続論理装置間で転送されるデータを一時記憶するキャッシュメモリ装置とを有する記憶システムにおいて、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、これらの装置に共用されるコモンバスにより相互に接続されるように構成する。

【0007】前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、いずれもモジュールで構成し、前記モジュールは、それぞれ、前記コモンバスに対し挿抜自在に取付けられるように構成する。

【0008】前記コモンバスは、ブラッタ上に配設され、前記上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記ブラッタに対し挿抜自在に取付けられるように構成する。

【0009】前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記コモンバスは、いずれも少なくとも二重化されており、前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記コモンバスの一方により縮退運転が可能となるように構成する。

【0010】前記二重化された上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置は、いずれも活線挿抜ができるように構成する。

【0011】前記記憶装置についても、同様に二重化された電源部を備えることができる。

【0012】前記記憶装置は、複数の小形記憶装置を組み合わせたアレ記憶装置で構成することができる。

【0013】前記キャッシュメモリ装置は、キャッシュメモリ本体を持ち、前記コモンバスに直接取り付けられるキャッシュメモリモジュールと、キャッシュメモリを持つ増設用のキャッシュユニットとを有しており、前記キャッシュユニットは、前記コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して接続されるように構成することができる。

【0014】前記上位側接続論理装置及び前記記憶装置側接続論理装置は、それぞれ、二重化されたマイクロプロセッサを有し、両マイクロプロセッサによりデータの比較チェックを行なうように構成することができる。

【0015】なお、コモンバス上には、上記上位側接続論理モジュールとは別の形式の上位側インタフェースや、上記記憶装置側接続論理モジュールとは別の形式の記憶装置側インタフェースを置き換えたり増設したりすることもできる。

【0016】

【作用】上記構成に基づく作用を説明する。

【0017】本発明によれば、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、これらの装置間で転送されるデータを一時記憶するキャッシュメモリ装置(複数の上位側接続論理装置及び複数の記憶装置側接続論理装置に共有されるキャッシュメモリ装置)とを有する記憶システムにおいて、前記複数の上位側接続論理装置、複数の記憶装置側接続論理装置、及びキャッシュメモリ装置は、これらの装置に共有されるコモンバスにより相互に接続されるように構成したので、上位側接続論理装置と記憶装置側接続論理装置とキャッシュメモリの増設または変更は、単にこれらをコモンバス上に追加しまたは変更して行くだけでよく、増設によるアップグレードが容易に達成できスケラブルなシステム構成を得ることができる。

【0018】また、これらの上位側接続論理装置、記憶装置側接続論理装置及びキャッシュメモリ装置は、モジュール化されて、コモンバスの配設されたブラッタに挿抜(着脱)自在に取り付けるようにしたので、これらの装置の必要な数量の増設作業も簡単である。

【0019】また、上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置、及びこれらの間を接続するコモンバスは、二重化され、2系統に分けて配線されているので、これらの装置の一方に障害が発生したときでも、他方の装置を用いて縮退運転が可能である。なお、障害発生時に縮退運転状況を示す情報は、共有メモリに書き込まれる。

5

【0020】この場合、上位側接続論理装置、記憶装置側接続論理装置、及びキャッシュメモリ装置は、いずれも活線挿抜対応のコネクタ部を具備しているので、システムを停止することなく保守点検を行なって故障部品の交換を行なったり、増設用の部品を追加したりすることが可能である。

【0021】電源部も二重化され、それにより無停電電源装置を実現する。

【0022】記憶装置は、複数の小形記憶装置を組み合わせたアレイド形とされ、これにより従来の大形ディスク装置1台を用いたものに比べてアクセスタイムを短縮できる。

【0023】キャッシュメモリ装置は、コモンバスに直接取り付けられるキャッシュメモリモジュール（キャッシュメモリパッケージ）と、増設用のキャッシュユニットとで構成され、増設用のキャッシュユニットは、コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して必要数接続されるようになっていて、簡単に増減することができる。

【0024】異常により、高信頼性の記憶システムを得ることができる。

【0025】

【実施例】以下に、本発明の実施例を図面の図1から図18により説明する。

【0026】図1は本発明の概念図を示す。図1により、本実施例の概要を説明する。

【0027】1は、対上位CPU（ホスト）接続用論理モジュールであるホストアダプタ部、2は、対記憶媒体接続用論理モジュールであるディスクアダプタ部、3は、両モジュール間で転送されるデータを一時記憶するキャッシュメモリパッケージ（キャッシュメモリモジュール）、4はホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3の間のデータ転送制御を司るコモンバス、5は、縦横にアレイド状に配置した記憶媒体である磁気ディスク群（以下「アレイドディスク」という）である。ホストアダプタ1は、上位インタフェース側のデータ形式及びアドレス形式を記憶媒体インタフェース用のデータ形式及びアドレス形式に変換する手段と、これらを制御管理する二重化したマイクロプロセッサとを有している。ディスクアダプタ2は、記憶媒体へデータを格納するためのアドレス演算機能と、記憶データ保証用冗長データの生成機能と、記憶媒体構成情報を認識する機能と、これらを制御管理するマイクロプロセッサとを有している。

【0028】図1において、上位装置（CPU）から送られてきた書き込みデータは、ホストアダプタ1からコモンバス4を介して一度キャッシュメモリパッケージ3に書き込むことにより上位に終了報告を行い、その後の空き時間でキャッシュメモリパッケージ3からディスクアダプタ2を経由してアレイドディスク5に書き込む。

6

【0029】また、上位装置からのデータ読み出し命令に対しては、キャッシュメモリパッケージ3上にデータが存在する場合はアレイドディスク5からは読み出さず、キャッシュメモリパッケージ3上のデータを上位装置に転送する。一方キャッシュメモリパッケージ3上にデータが存在しない場合は、アレイドディスク5からディスクアダプタ2によりコモンバス4を経由して一度キャッシュメモリパッケージ3に書き込まれた後同様にホストアダプタ1を経由して上位装置へ転送する。

10 【0030】コモンバス4上のホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3各々はその接続数を任意に変えることができる。ホストアダプタ1の実装数を変えれば対上位接続バス数が増え、上位ホストに対するデータ転送能力を高めることができる。ディスクアダプタ2の実装数を変えれば記憶媒体に対する接続バス数が増え、記憶媒体に対するデータの書き込み/読み出しの転送能力を高めることができる。また、同時に記憶媒体の数も増加することができる。キャッシュメモリパッケージ3の実装数を変えればデータの一時格納場所であるキャッシュメモリの容量が増え、記憶媒体の総容量に対するキャッシュメモリの容量の比率を高めることができるので、対上位装置からアクセスするデータがキャッシュメモリ上に存在する確率（以下「キャッシュヒット率」という）を高める等スケラブルな装置構成を実現できる。

【0031】図2は、図1の概念図の詳細な構成図を示したものである。図2は、図1の複数台のホストアダプタ及び複数台のディスクアダプタのうち、それぞれ1台だけを示し、他は図示を省略している。

30 【0032】ホストアダプタ1において、6はホストインタフェースの光信号を電気信号に変換する信号変換部、7は上位データフォーマットをアレイドディスク5用フォーマットに変換するフォーマット変換部である。8はコモンバス4とのデータの授受を司るデータ転送制御部で、内部にバケット転送単位のデータを格納する記憶バッファを内蔵している。9は活線挿抜対応可能な小振幅電流駆動形バスドライバ（以下「BTL」という）である。

40 【0033】ホストからのデータ転送要求は10のマイクロプロセッサ（以下「MP」という）に引継がれ、ホストアダプタ1内のデータ転送制御は当MP10の管理下で行われる。

【0034】MP10はMP内の障害発生を検出するなど高信頼性を確保するために2重化されており、11のチェック部で同じ動作をする2重化されたMP10とMP10'を比較チェックしている。

50 【0035】12はMP10の制御プログラムを格納するブートデバイスで、このブートデバイス12には書き換え可能な大容量フラッシュメモリを採用しており、またMP10は必要に応じて13のローカルメモリに制御

プログラムをコピーして使用することにより、MP10のメモリアクセス時間の高速化を実現しており、図中破線で囲まれた部分29がチャンネルアダプタモジュールであり、ホストアダプタ1には当モジュール29が2回路搭載してある。

【0036】ディスクアダプタ2において、14はアレイドディスクに書き込むデータをセクタ単位に格納するバッファメモリ、15はバッファメモリ14の制御及びデータ転送制御を行なうデータ制御バッファ部、16はアレイドディスク5に書き込むデータを保証するための冗長データを生成する冗長データ生成部、17はアレイドディスク5(ターゲット)に対するイニシエータ(SCSIのマスタ側インタフェース)である。

【0037】またディスクアダプタ2内のデータ転送制御は、ホストアダプタ1と同じ構成をとるMP周辺部(MP10、MP10'、チェック11、ブートデバイス12、ローカルメモリ13からなりディスクアダプタ用の制御プログラムを搭載する)の管理下で行なわれる。

【0038】アレイドディスク5は、図2では4つのディスク(ターゲット)しか示していないが、実際には1台のディスクアダプタ2に対し例えば4(横)×4(縦)~4(横)×7(縦)つのディスクで構成される。横列はECCグループ(Error Correction Group)を構成し、各ECCグループは例えば3つのデータディスクと1つのパリティディスクで構成される。更に、後述のように、このようなアレイドディスク5の1組に対し、二重化されたホストアダプタと二重化されたホストアダプタと二重化されたディスクアダプタを通じて、あるCPUからアクセスできるようになっている。そして、ホストアダプタの一方に障害が発生したときには、ホストアダプタの他方もしくはディスクアダプタの他方を通じて、同じCPUから同じアレイドディスクにアクセスすることができる。

【0039】キャッシュメモリパッケージ3において、18は各アダプタのMP10が共通にアクセス可能で種々の管理情報を記憶する共有メモリ部、19は共有メモリ制御部、20はキャッシュメモリ部、21はキャッシュメモリ制御部であり、両メモリ制御部19、21は共にメモリ書き込みデータ保証のためのECC生成回路、読み出しデータの検査及び訂正回路を内蔵し、キャッシュメモリパッケージ3全体で最大1GBのキャッシュ容量を実現しており、装置構成上は2面化して実装している。

【0040】キャッシュメモリ容量を更に増設する場合は、キャッシュメモリパッケージ3の代わりに(または、キャッシュメモリパッケージ3に加えて)22で示すキャッシュポートパッケージを実装し、23で示すブラッタ(基板差し込み板)間接続ケーブルを介して24で示すキャッシュユニットに接続し、(すなわち、増設

ユニット24内のキャッシュメモリには、キャッシュポートパッケージ22及びケーブル23を介してアクセスできるように構成され)、これによって、最大8GB2面までキャッシュ容量を増設することができる。図2では、キャッシュメモリパッケージ2を2面設けたのに加えて、キャッシュポートパッケージ22を実装し、これにケーブル24を介していくつかのキャッシュユニット24を接続した場合を示している。

【0041】以上述べたホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3はコモンバス4を介してつながっているが、このコモンバス中、25は各アダプタのMP10が共有メモリをアクセスするためのマルチプロセッサバス(以下「Mバス」という)、26は高速データ転送を行う高速I/Oバス(以下「Fバス」という)である。

【0042】高速I/Oバス26は通常は64ビット幅で2系統同時に動作しているが、障害発生時はどちらか1系統のみでの縮退動作が可能であり、またMバス25に障害が発生した場合はFバス26のどちらか1系統を使用して動作可能である。

【0043】更に活線挿抜対応(挿抜の際、挿抜部品の負荷を小さくして挿抜を行なうことで、システムを稼働状態のまま挿抜を可能とする)のBTL9をコモンバス4のインタフェースにすることで、ホストアダプタ1に障害が発生した場合、システムは自動的に本障害バスを閉塞し他のホストアダプタのバスを用いてアレイドディスク5に対し対上位(同じCPU)からのアクセスを継続する。保守員は、システム稼働状態において障害の発生したホストアダプタ1を取り除き、正常なホストアダプタ1をシステムに挿入し、27の保守用プロセッサ(以下「SVP」という)から28のLANを介して復旧の指示を与え、システムは交換されたホストアダプタ1の動作をチェックし正常であれば閉塞バスを復旧させることにより、無停止運転を実現している。なお、図中LANCは、LAN Controller(SVPインタフェースコントローラ)である。SVP27は、他のホストアダプタ及びディスクアダプタにも同様に接続され、監視及び保守が行なわれるようになっている。

【0044】また、各アダプタの制御プログラムに変更がある場合は、SVP27からLAN28を介してブートデバイス12内にある制御プログラムの内容を書き替えることにより無停止のアップグレードが可能である。

【0045】即ち、システムの制御プログラムをアップグレードを実施する場合は、まずホストアダプタ/ディスクアダプタの各モジュールを1モジュールずつ閉塞し、制御プログラムのアップグレードを行い再接続する。以上のように1モジュールずつの制御プログラムの入れ換え操作を繰り返すことにより、系全体の制御プログラム入れ換えが実施される。

【0046】図3は、図2に示した構成図に沿ってデー

タの流れとデータの保証を示した図である。

【0047】上位からアレイドィスクにデータを書き込む場合、例えばESCON（光チャネルの商標名、IBM社）から、先ず書き込み先の記憶空間上の物理アドレス情報（以下「PA」という）が送られて来た後、データ（CKD（Count Key Data）フォーマット）+CRCコードが送られてくる。これらの光信号は信号変換部6で電気信号に変換すると共にパリティを生成し、フォーマット変換部7ではデータフォーマットをFBA（Fired Blocked Architecture）フォーマットに変換すると共にLRC（Longitudinal Redundancy Check、長手方向冗長度チェック）コードを付加し、更にPAをデータの一部として取り込んでアレイドィスク上の論理アドレス（以下「LA」という）を生成した後これら総ての情報に対してパリティを付加してFバス26に送られる。

【0048】キャッシュパッケージ3では、Fバス26からのデータに対して誤り訂正可能なECCを付加してキャッシュメモリ20に書き込む。

【0049】ディスクアダプタ2では、Fバスからのデータに対して更にCRCコードが付加され、該データSCSIインターフェースを介してアレイドィスク5に送られ、磁気ディスク装置個々にECCを付加して書き込みデータを保証している。

【0050】アレイドィスク5からのデータ読み出しにおいても同様に、各チェックコードを元に読み出しデータの検査/訂正を行い信頼性を高めている。

【0051】以上のように、チェックコードはデータの長さ方向に対してはある長さ毎の水平チェック、データの垂直（幅）方向に対しては（例えばバイト単位の）垂直チェックで2重化されており、また転送が行われる領域間（図中一点鎖線）では当該2重化チェックコードのうち1つを必ずデータとして受け渡すことによりデータ保証に万全を期している。

【0052】図4は図1で述べたスケラビリティを実現するための装置外観図であり、41はアレイドィスクを制御する制御ユニット部、42はアレイドィスクを実装するアレイドィスク部で、本装置はこの2つのユニットで構成される。

【0053】図5は制御ユニット41の実装図で（a）は正面図、（b）は側面図を表わす。51はホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3を実装する論理架部、52は停電時に揮発メモリであるキャッシュメモリ部に電源を供給するバッテリー部、53はキャッシュメモリ増設時にキャッシュユニット24及び増設メモリ用の追加バッテリーを実装するキャッシュメモリ増設部、54はSVP実装部、55は論理架に電源を供給する論理架用スイッチング電源、56はアレイドィスクの構成（容量）が小規模の場合のアレイ

ディスク実装部、57はアレイドィスク部に電源を供給するアレイドィスク用スイッチング電源、58は両スイッチング電源55、57に電源を供給する商用電源制御部である。

【0054】図6は大容量アレイドィスクを構成するときのアレイユニット部の実装図で（a）は正面図、（b）は側面図を表わす。

【0055】アレイドィスク実装部56は、磁気ディスク装置を最大112台（8行x7列x2）実装可能であり、各磁気ディスク装置に障害が発生した場合の装置の入れ替えを容易にするために、装置の正面と背面の両面から挿抜可能となるような実装方式をとっている。

【0056】61はユニット全体の発熱を逃がすための冷却ファンで、冷却効果を高めると共に、騒音抑止の観点から小さな冷却ファンを使って小区分化し、床面より天井へ送風する構造をとっている。

【0057】図7は図5で説明した論理架部の接続方式図である。

【0058】71はコモンバス4をプリント配線したブラッタ（基板の押し込み用の板）であり、72は各アダプタ、パッケージとブラッタ71を接続するためのコネクタである。

【0059】ホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3の間のデータ転送はコモンバス4を介して行うため、各アダプタ、パッケージはコネクタ72上の任意のどの位置でも接続可能となり、ホストアダプタ1の実装数、ディスクアダプタ2の実装数を自由に変えることができる。

【0060】一方、キャッシュ容量を増設する場合はキャッシュメモリパッケージ3をキャッシュポートパッケージ22に変えて実装するか、または図7に示すように、キャッシュメモリパッケージ3に加えてキャッシュポートパッケージ21を実装し、これに、接続ケーブル23を介してキャッシュユニット43（図2の24に相当）に接続することにより、もとの2GBの容量に加えて更に最大8GB2面分のキャッシュメモリ容量を拡張できる。

【0061】図8は図5で示した論理架部の実装イメージ図である。

【0062】図8で、コモンバス4は、ブラッタ71上を左右方向にプリント配線されており、このブラッタ71に対して、キャッシュポートパッケージ22の基板（CP）の取付部、キャッシュメモリパッケージ3の基板（C）の取付部、ホストアダプタモジュールの基板（H）の取付部、及びディスクアダプタモジュールの基板（D）の取付部が設けられ、図の矢印84で示すように、各基板は、挿抜操作面側から着脱されるようになっていて、ブラッタ71に差し込まれるとコモンバス4と電気接続されるものである。

【0063】81は、ホストアダプタ1の基板上の下方

部に実装されて、対上位インターフェイスを司る光コネクタ部、82はディスクアダプタ2の基板上の下部に実装されて、アレイディスク5と接続するSCSIコネクタ部、83はキャッシュポートパッケージ22を実装したときの接続ケーブル23用の接続コネクタ部である。85は、キャッシュメモリパッケージ3の基板(C)の下部に取付けたキャッシュメモリ本体(図2のキャッシュメモリ20)である。

【0064】各コネクタ部は、障害発生等で各アダプタ、パッケージを挿抜する際の操作性を向上させるため、接続コネクタ83を除き、操作面84側へは実装せず、ブラッタ71の接続側に集中実装している。

【0065】図9は本発明のソフトウェア構成を示した図である。

【0066】91はホストアダプタ1のブートデバイス12に書き込まれるチャンネルアダプタ制御プログラム(以下「CHP」という)、である。また、ディスクアダプタ2のブートデバイス12に書き込まれるディスクアダプタ制御プログラムのうち、92はアレイディスク固有の処理およびキャッシュメモリとアレイディスク間のデータ転送制御を受け持つディスクアダプタマスタ制御プログラム(以下「DMP」という)、93はDMP92の制御管理下でキャッシュメモリ20とアレイディスク5の間のデータ転送制御を受け持つディスクアダプタスレーブ制御プログラム(以下「DSP」という)である。

【0067】ディスクアダプタ2のブートデバイス12には、DMP92とDSP93の2種類が書き込まれているが、装置構成上nセットのディスクアダプタでアレイディスクにアクセスする場合、そのうちの2セットがDMP92として動作(2重化)し、残るn-2のディスクアダプタがDSP93として動作する。

【0068】94はSVP27に搭載するSVP制御プログラムで、CHP91、DMP92、DSP93を監視及び保守するとともに、各制御プログラムの更新時はSVP27から更新したいMPの制御プログラムを直接、または他のMPから当該MPの制御プログラムを更新することができる。

【0069】図10はデータの流れに基づいた図9で示したソフトウェア構成の機能分担を示した図である。

【0070】CHP91は、上位からのアドレス形式及びデータ形式を下位アドレス形式及びデータ形式に変換し、キャッシュメモリに書き込む。101はセグメント、102はブロック、103はアレイディスク5上の磁気ディスク1台当りに書き込むデータ量を表すストライプである。DMP92は、キャッシュメモリ上からストライプ単位にデータを読み出し、下位アドレスをアレイディスクの行NO、列NO、FBA、ブロック数に変換し、DSP93でアレイディスクにデータを書き込む。

【0071】また、DMP92はアレイディスク5の構成情報も管理している。

【0072】以上のように、各制御プログラムを機能分担することにより、上位インタフェースをSCSIやファイバーチャネル等に変更する場合はCHP91のみ、またアレイディスク構成を変更(ディスクの行数/列数、RAID(Redundant Array Inexpensive Disk)方式等)する場合はDMP92のみの変更で対応可能であり、ホストアダプタ1、ディスクアダプタ2の接続変更に合わせて各制御プログラムを書き替えることで、スケールビリティを実現するとともに、ソフトウェア開発の負荷も軽減している。

【0073】図11はコモンバス4の2重化の考え方や縮退動作を説明した図である。

【0074】111はコモンバス4の使用権を獲得することのできるバスマスタ(MP10を搭載しているホストアダプタ1又はディスクアダプタ2)、112はバスマスタ111からのアクセス要求を受けるバスマスレーブ(キャッシュメモリパッケージ)である。

【0075】Fバス26は通常動作状態では64ビットバス(200MB/S)2系統を同時に動作させ400MB/Sを実現しており、各バス系統はパリティチェック又はタイムアウトで障害を検出可能である。障害発生時はバスマスタ111は各自縮退状態に入り、残る1系統を使ってバスマスレーブをアクセスすると共に、この時の縮退情報は共有メモリ18上の管理エリアに登録される。

【0076】またコモンバス内のシステム制御信号(バスリセット等)は信号線を3重化しており、通常動作時は3線一致、縮退動作時は2線一致(多数決)方式を採用することにより信頼性を高めている。

【0077】図12は装置各部位における多重化と縮退運転を示した図である。

【0078】121は2ポート化されたチャンネルバスであり、ホストアダプタ1にはチャンネルアダプタ29が2モジュール、対上位用のチャンネルバスが4バス実装しており、障害発生時は交替チャンネルアダプタ(CHP)、交替チャンネルバスを使用して縮退運転に入る。

【0079】122はディスクアダプタ2とアレイディスク5の間のインタフェースを司るSCSIバスで、1行の磁気ディスク群に対して別のディスクアダプタ2からもアクセス可能なように2重化しており、当バスに障害が発生した場合は交替SCSIバスを使用して縮退運転に入る。また、アレイディスクマスタ制御を行うDMP92も2重化しており、障害発生時は交替DMP92を使用して縮退運転に入る。

【0080】共有メモリ18、キャッシュメモリ20も2重化しており、共有メモリに障害が発生した場合は残るもう一方の使用して縮退運転に入り、キャッシュメモ



りに障害が発生した場合はライトペンディングデータ（キャッシュメモリ上に残っているデータ）をディスクにデステージし障害発生メモリ部位を除いたメモリで縮退運転を行う。

【0081】アレイディスク5上の磁気ディスクに障害が発生した場合は、当該磁気ディスクを切り離し予備の磁気ディスクに修復しながら読み出し書き込み動作を行う。

【0082】図13は装置の電源系の多重化と縮退運転を示した図である。

【0083】商用電源制御部58は各々独立したAC入力で2重化して、論理架用スイッチング電源55及びアレイディスク用スイッチング電源57にそれぞれ供給しているため、障害発生時はもう片方の商用電源制御部58で縮退運転に入る。

【0084】131は上位ホストからの電源ON/OFFの遠隔制御や商用電源制御部58、両スイッチング電源等の電源回路を制御する電源制御回路（以下「PC1」という）である。

【0085】論理架用スイッチング電源55は冗長運転用として必要数より2回路多く実装し電源コモンバスを介して論理架51及びバッテリー52に供給することにより、当スイッチング電源55が2回路故障しても動作可能である。

【0086】同様に列単位の磁気ディスク群に供給するアレイディスク用スイッチング電源57も、冗長運転用として2回路多く実装し電源コモンバスを介して供給することにより、当スイッチング電源57が2回路故障しても動作可能であり、さらに両スイッチング電源55、57を2重化するよりも安価な構成に仕上げることができる。

【0087】また停電時においては、2重化されたバッテリー52から電源コモンバスを介して論理架内の揮発メモリであるキャッシュメモリ及びPC1131に供給され、片方のバッテリーが故障しても動作可能である。

【0088】図14及び図15はアレイディスクに使用する磁気ディスク装置単体の記憶容量別にアレイディスクを構成したときのシステム性能を比較した図である。

【0089】図14はそれぞれ異なる磁気ディスク装置を使用して同一容量のアレイディスクを実現した場合の構成を示しており、項番141が3GBの磁気ディスク装置（3.5インチ径のディスクを使用）、項番142が4.0GBの磁気ディスク装置（5インチ径のディスクを使用）、項番143が8.4GBの磁気ディスク装置（6.4インチ径のディスクを使用）を使用している。アレイ構成は、ディスク装置141が14枚のデータディスクの2枚のバリティディスク、ディスク装置142が14枚のデータディスクと4枚のバリティディスク、ディスク装置143が14枚のデータディスクと2枚のバリティディスクで構成した場合である。

【0090】図15は各磁気ディスク装置141、142、143についての毎秒当りのI/O命令発行件数と平均応答時間の関係を示しており、アレイディスクシステムとしてのトランザクション性能を向上させるためには、小容量（小径）の磁気ディスク装置を使用してアレイ構成を大きくすることが最も性能を引き出せることから、本発明に於ては3.5インチ磁気ディスク装置141を採用してアレイディスクシステムを実現している。従って、同じ記憶容量の磁気ディスク装置を、従来のように大形磁気ディスク装置1台で構成するのと、複数台の小形磁気ディスク装置のアレイで構成するのとでは、後者の小形磁気ディスク装置を多数用いたアレイ構成のものの方が、平均アクセスタイムを短縮できる点で有利である。

【0091】以上説明してきたスケラブルなアーキテクチャを使用して実現できる装置モデル構成例を図16～図19に示す。

【0092】図16は、コモンバス4上のディスクアダプタ2の実装数を減らし、更にキャッシュポートバッケージ22を実装し、接続ケーブル23を介してキャッシュユニット24に接続することにより、キャッシュヒット率の高める高性能大容量キャッシュメモリ付小形ディスクアレイを実現した時の構成図である。

【0093】またディスクアダプタ2を実装しないで、ホストアダプタ1とキャッシュメモリのみで構成した場合（図中の破線内の構成）は、記憶媒体が磁気ディスクから半導体メモリに代わり、更に高速データ転送可能な高性能の半導体ディスク装置を実現する。

【0094】図17はディスクアダプタ2を最大構成とし、キャッシュバッケージ3を実装し又はキャッシュポート22を実装し接続ケーブル23を介してキャッシュユニットを接続することにより、高性能大容量キャッシュメモリ付大形ディスクアレイを実現した時の構成図である。

【0095】図18はホストアダプタ1の対上位インターフェースをSCSI/ファイバーチャネル等のインターフェースに変えて、ディスクアダプタ2の実装数を減らし、更にFバス26のビット幅を半分縮小した2系統で構成することにより、オープン市場をターゲットにした無停止運転の高性能フォールトトレラント（高信頼性）サーバシステムを実現した時の構成図である。

【0096】図19は図18の構成を元に2重化、活線揮散を考慮せずに、最もシンプルな構成をとることによって安価なオープン市場向けのサーバシステムを実現した時の構成図である。なお、図中、4D+1Pは、データディスク4枚とバリティディスク1枚の趣旨である。

【0097】以上の実施例において、コモンバス4上に、更に光ディスクアダプタ（光ディスク用接続論理モジュール）を介して光ディスク装置を接続し、磁気テープ制御装置（磁気ディスク接続論理モジュール）を介し

て磁気テープ装置を接続し、あるいは半導体記憶装置接続論理モジュールを介して半導体記憶装置を接続することができる。また、コモンバス4上に別の形式のホストアダプタを介してワークステーションを接続することもできる。このように、コモンバス上に、種々の形式の記憶装置に対する記憶媒体アダプタを接続することができる。

【0098】

【発明の効果】以上詳しく説明したように、本発明によれば、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、これらの装置間で転送されるデータを一時記憶するキャッシュメモリ装置（複数の上位側接続論理装置及び複数の記憶装置側接続論理装置に共有されるキャッシュメモリ装置）とを有する記憶システムにおいて、前記複数の上位装置側接続論理装置、複数の記憶装置側接続論理装置、及びキャッシュメモリ装置は、これらの装置に共有されるコモンバスにより相互に接続されるように構成したので、上位側接続論理装置と記憶装置側接続論理装置とキャッシュメモリの増設または変更は、単にコモンバス上にこれらの装置等を追加または変更して行くだけでよく、増設によるアップグレードが容易に達成できスケラブルなシステム構成を得ることができる。また、これらの上位側接続論理装置、記憶装置側接続論理装置及びキャッシュメモリ装置は、モジュール化されて、コモンバスの配設されたブラケットに挿抜（着脱）自在に取り付けるようにしたので、これらの装置の必要な数量の増設作業も簡単であるという効果がある。

【0099】また、上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置、及びこれらの間を接続するコモンバスは、二重化され、2系統に分けて配線されているので、これらの装置の一方に障害が発生したときでも、他方の装置を用いて縮退運転が可能である。この場合、上位側接続論理装置、記憶装置側接続論理装置、及びキャッシュメモリ装置は、いずれも活線挿抜対応のコネクタ部を具備しているので、システムを停止することなく保守点検を行なって故障部品の交換を行ったり、増設用の部品を追加したりすることが可能であるという効果がある。

【0100】更に、記憶装置は、複数の小形記憶装置を組み合わせたアレイ形とされ、これにより従来の大形ディスク装置1台を用いたものに比べてアクセスタイムを短縮できるという効果がある。

【0101】また、キャッシュメモリ装置は、コモンバ

スに直接取り付けられるキャッシュメモリモジュール（キャッシュメモリパッケージ）と、増設用のキャッシュユニットとで構成され、増設用のキャッシュユニットは、コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して必要数接続されるようになっているので、簡単に増減することができるという効果も得られる。

【0102】以上により、高信頼性の記憶システムを得ることができる。

【図面の簡単な説明】

【図1】本発明の実施例の概要を示す概念図である。

【図2】本発明の一実施例の記憶システムの詳細な構成図である。

【図3】図2の構成図に沿ったデータの流れとデータ形式を示した図である。

【図4】本発明の一実施例の装置外観図である。

【図5】本発明の一実施例の装置における制御ユニット部の実装方式図である。

【図6】本発明の一実施例の装置におけるアレイディスクユニット部の実装方式図である。

【図7】本発明の一実施例の装置における論理架部の接続方式図である。

【図8】本発明の一実施例の装置における論理架部の実装方式図である。

【図9】本発明の実施例に適用されるソフトウェア構成図である。

【図10】本発明の実施例によるデータの流れとソフトウェアの機能分担を示した図である。

【図11】本発明の実施例によるコモンバスの2重化と縮退動作を示した図である。

【図12】本発明の実施例による装置各部位の2重化と縮退運転を示した図である。

【図13】本発明の実施例による装置の電源系の多重化と縮退運転を示した図である。

【図14】アレイディスクに使用する磁気ディスク装置単体のディスク構成を示す図である。

【図15】磁気ディスク装置の記憶容量とアレイディスクのシステム性能を示した図である。

【図16】高性能大容量キャッシュメモリ付小形ディスクアレイの構成図である。

【図17】高性能大容量キャッシュメモリ付大形ディスクアレイの構成図である。

【図18】高性能フォールトトレラントサーバシステムの構成図である。

【図19】低価格サーバシステムの構成図である。

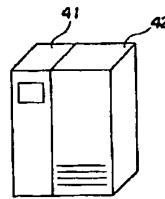
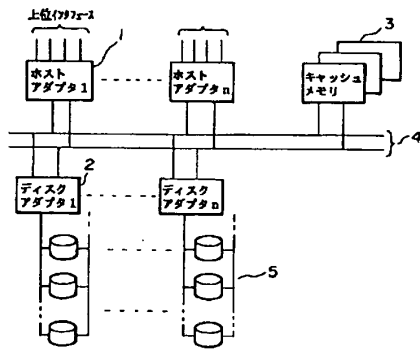
【図20】従来の記憶システムの概略構成図である。

【図1】

【図4】

【図1】

【図4】

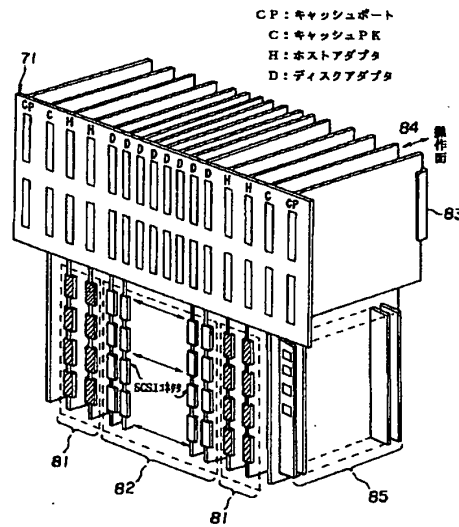
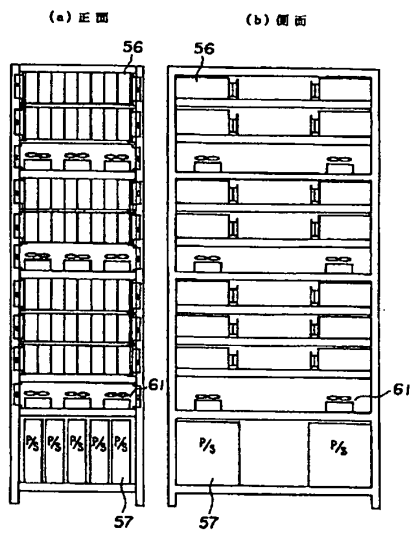


【図6】

【図8】

【図6】

【図8】

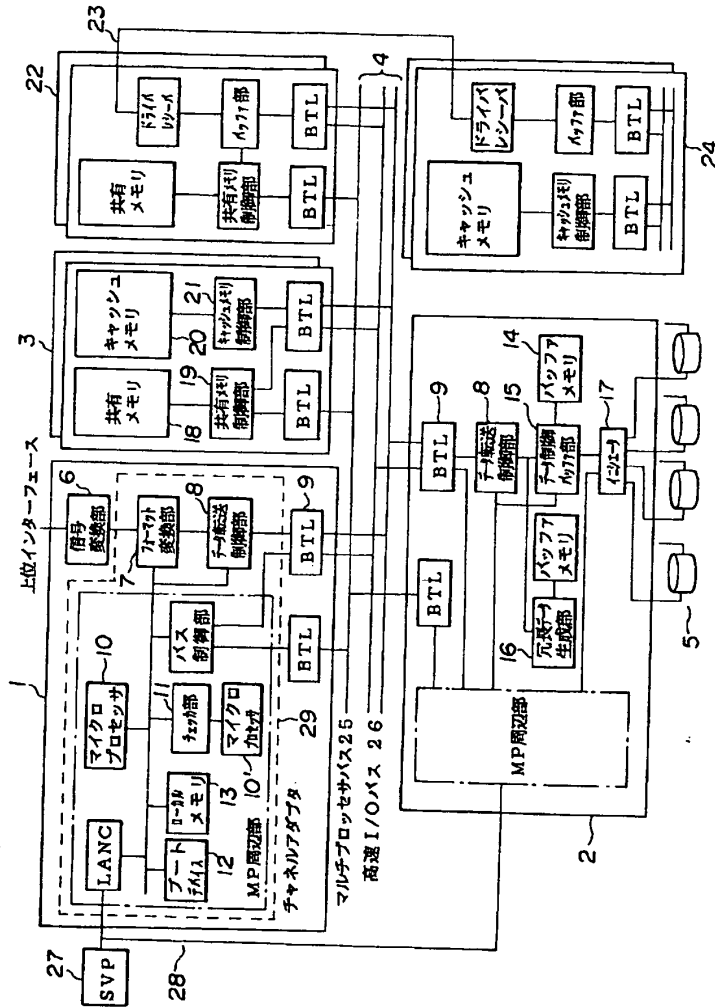


(11)

特開平7-20994

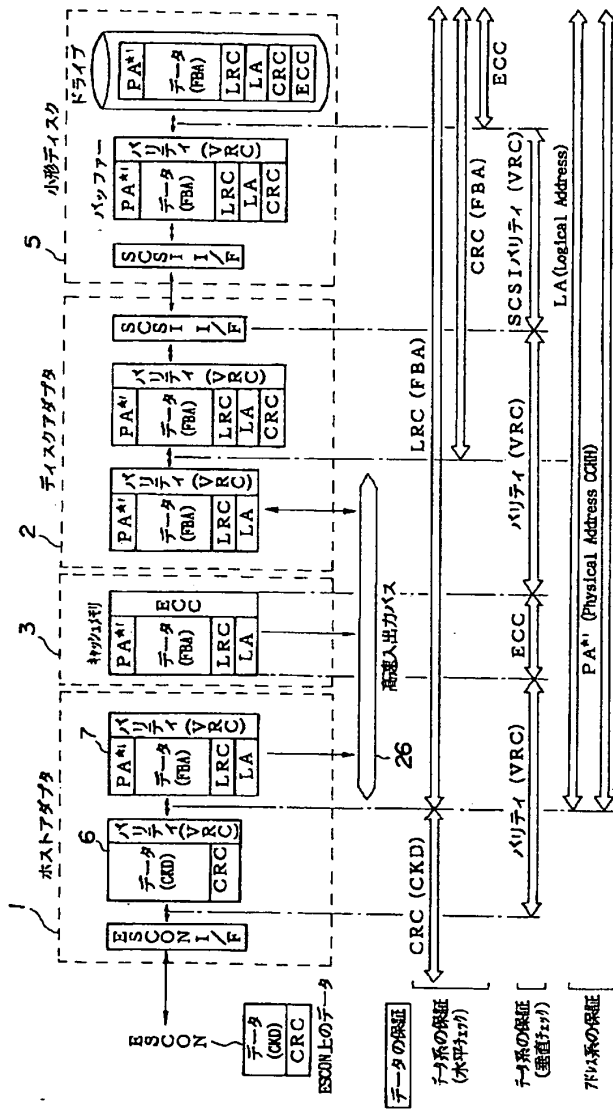
【図2】

【図2】



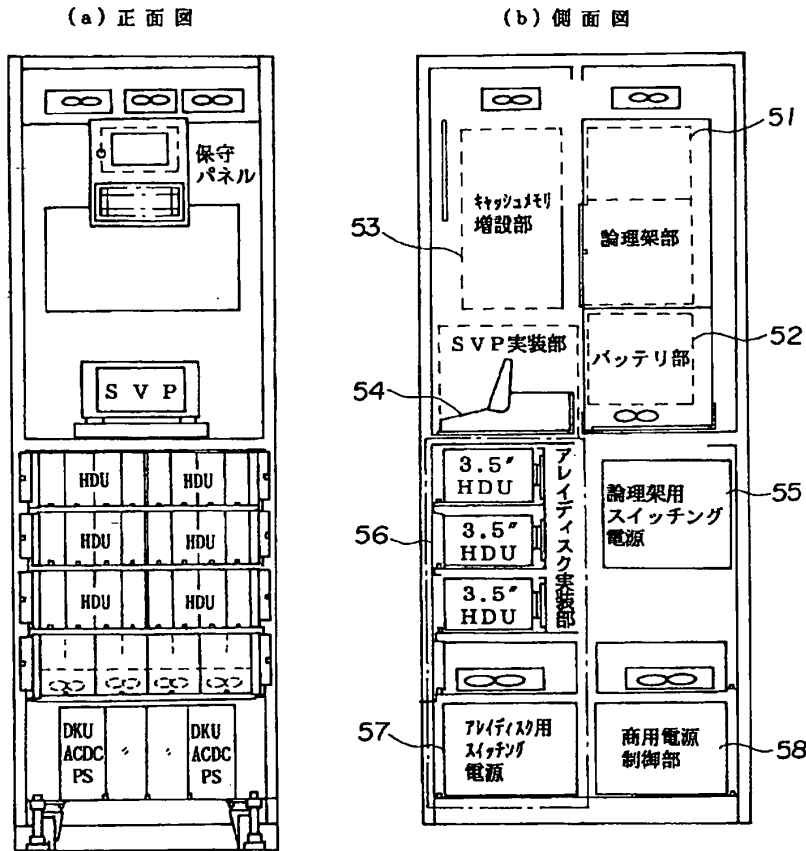
【図3】

【図3】



【図5】

【図5】



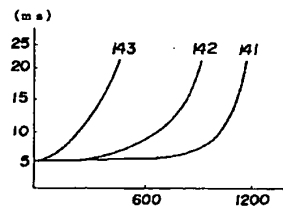
【図14】

【図15】

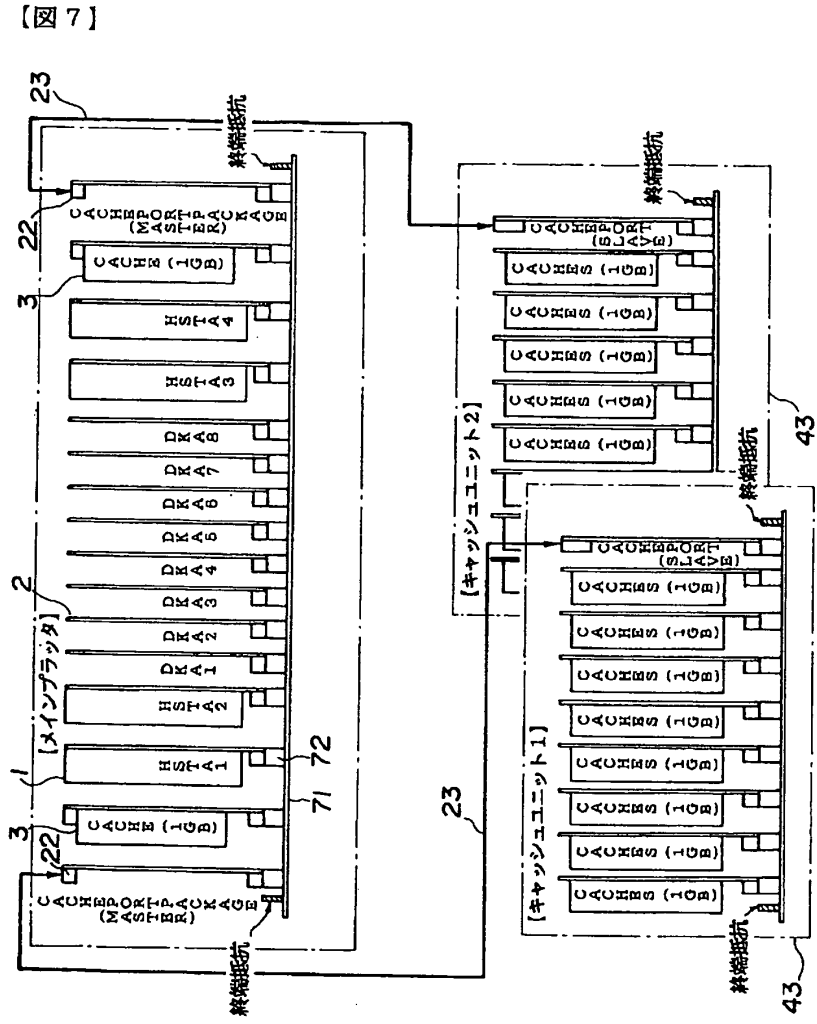
【図14】

【図15】

項番	磁気ディスク単体容量	アレイ構成	アレイ容量
141	3.0GB (3.5インチ)	(14D+2) × 5	約220GB
142	4.0GB (5インチ)	(14D+2) × 4	
143	8.4GB (6.4インチ)	(14D+2) × 2	

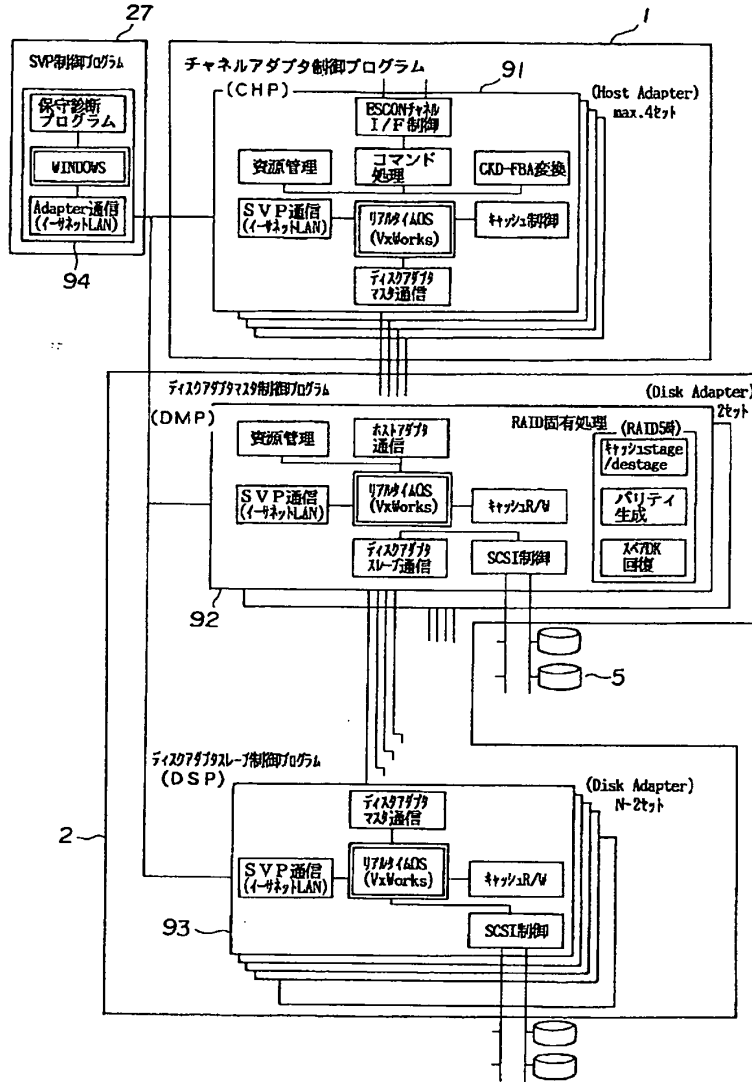


【図7】



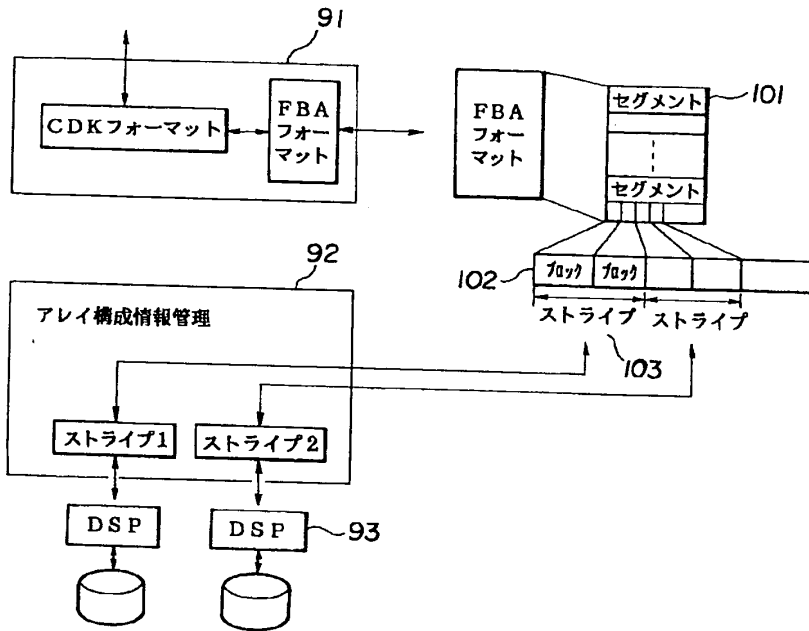
【図9】

【図9】





【図10】



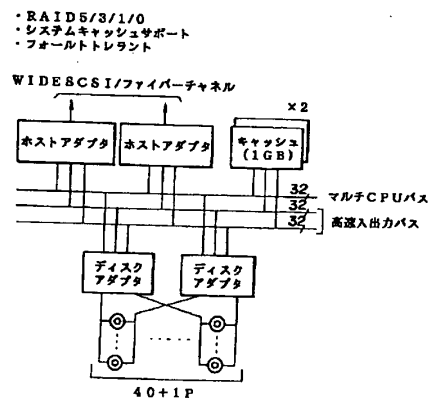
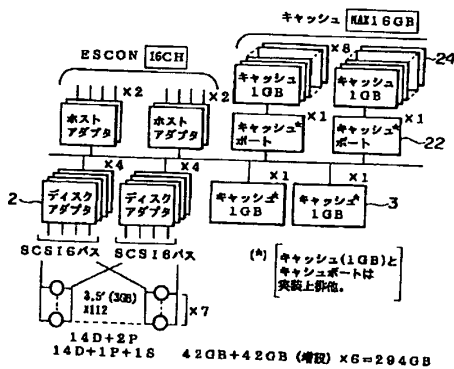
【010】

【図17】

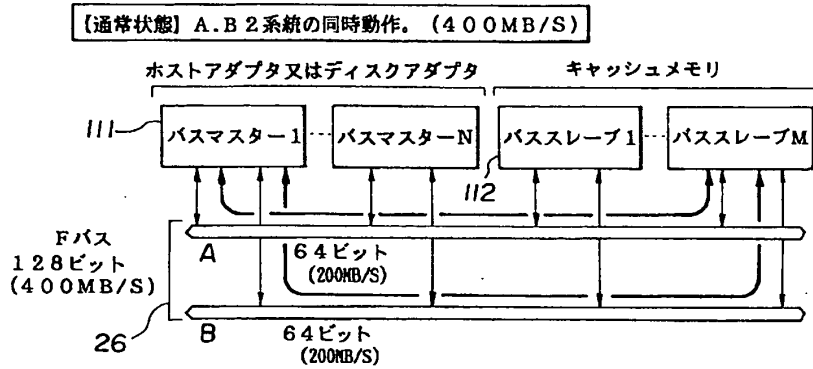
【図18】

【図17】

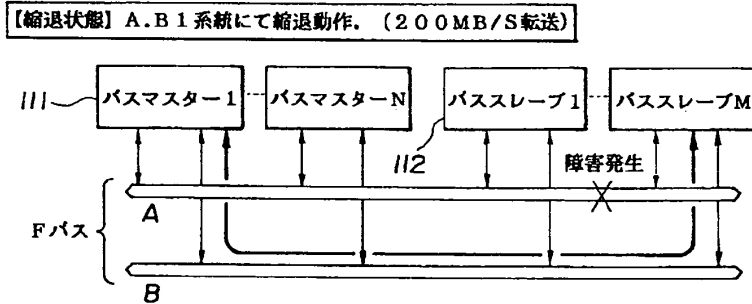
【図18】



【図11】

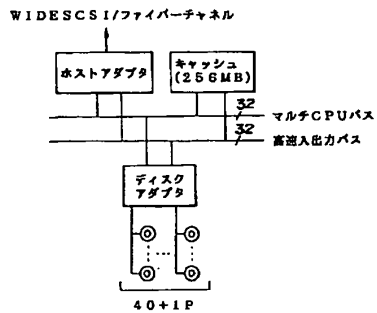


【図11】

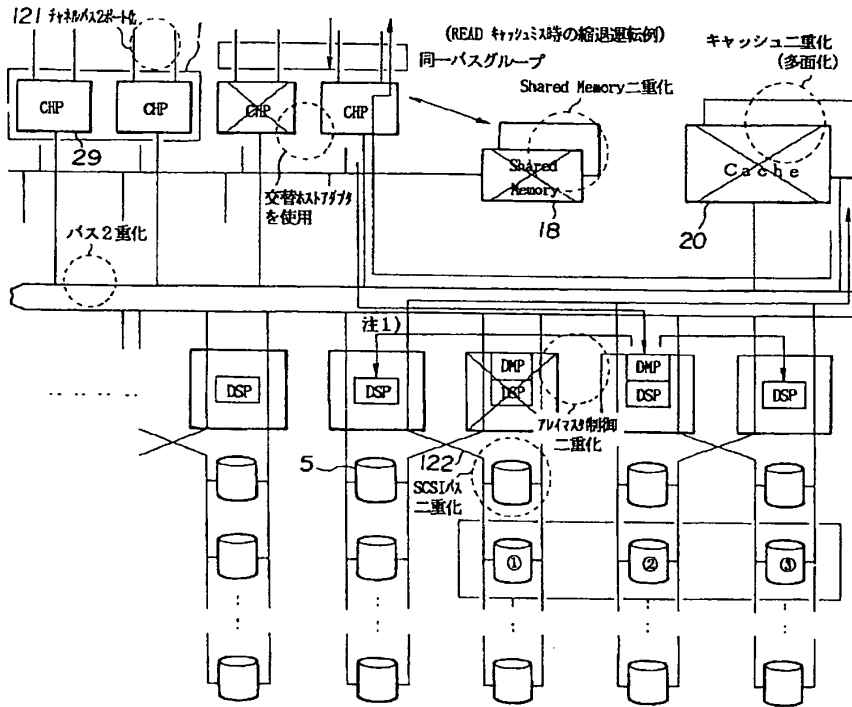


【図19】

【図19】

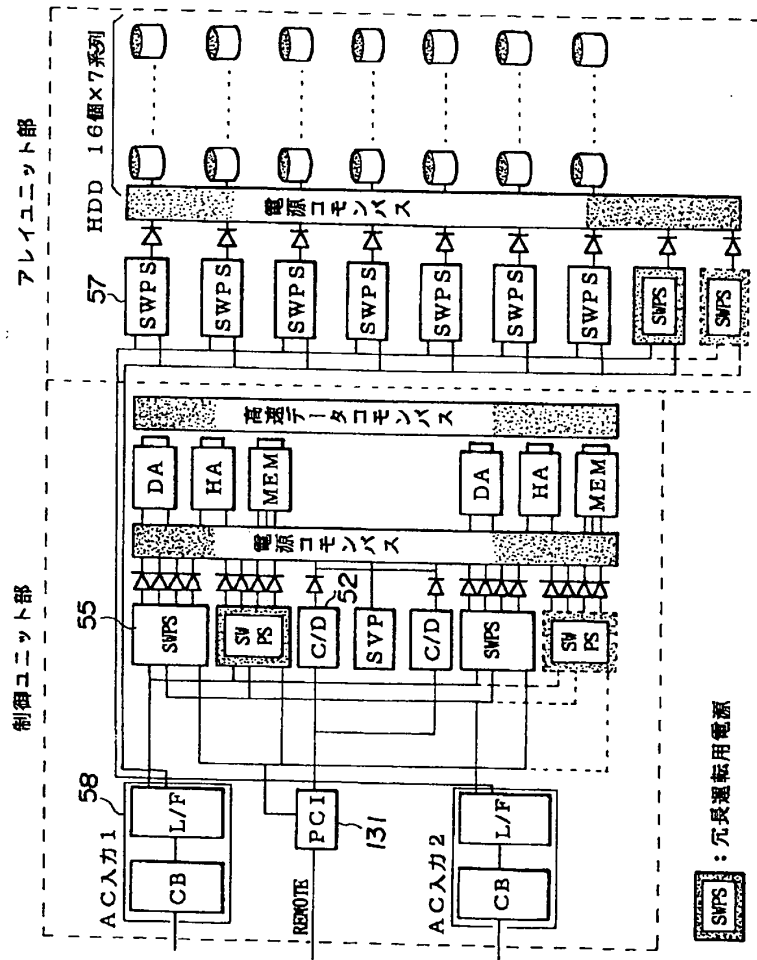


【図12】



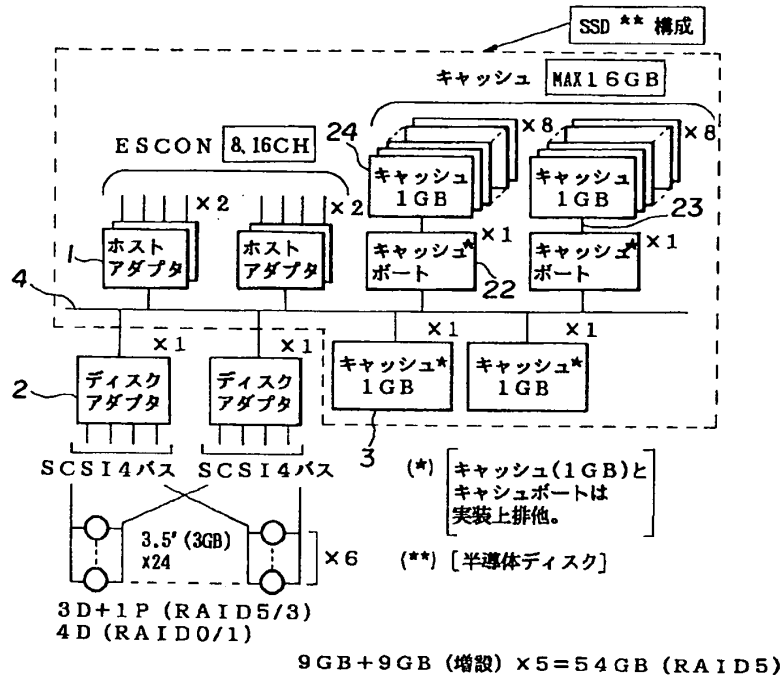
【図13】

【図13】

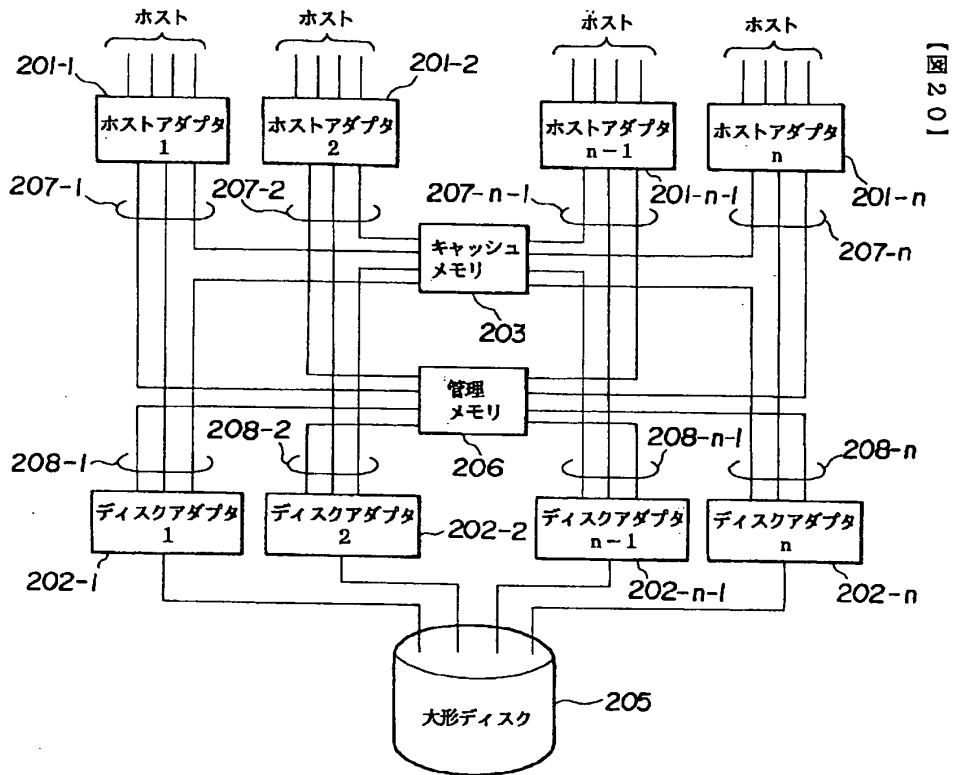


【図16】

【図16】



【図20】



フロントページの続き

(72)発明者 高橋 直也  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 井上 靖雄  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 岩崎 秀彦  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 星野 政行  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 磯野 聡一  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

【公報種別】特許法第17条の2の規定による補正の掲載

【部門区分】第6部門第3区分

【発行日】平成10年(1998)9月25日

【公開番号】特開平7-20994

【公開日】平成7年(1995)1月24日

【年通号数】公開特許公報7-210

【出願番号】特願平5-162021

【国際特許分類第6版】

G06F	3/06	301
	12/08	320
	13/12	330

【F I】

G06F	3/06	301 B
	12/08	320
	13/12	330 T

【手続補正書】

【提出日】平成9年2月5日

【手続補正1】

【補正対象書類名】明細書

【補正対象項目名】特許請求の範囲

【補正方法】変更

【補正内容】

【特許請求の範囲】

【請求項1】 少なくとも1つの上位装置に接続され、前記上位装置に対するインターフェースを構成する複数の上位側接続論理装置と、上位装置から転送される情報を記憶する記憶装置と、前記記憶装置に接続され、前記記憶装置に対するインターフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置との間で転送されるデータを一時記憶する二重化されたキャッシュメモリ装置と、前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置とに接続され、前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置との間のデータ転送を行う1組のバスとして動作する2系統のデータ転送用バスを含む共通バスと、  
から構成される記憶システム。

【請求項2】 請求項1に記載の記憶システムにおいて、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置はモジュールで構成され、前記モジュールの各々は、それぞれ前記共通バスに対し挿抜自在に取り付けられることを特徴とする記憶システム。

【請求項3】 請求項1に記載の記憶システムにおいて、

前記共通バスはブラケット上に配設され、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記ブラケットに対し挿抜自在に取り付けられることを特徴とする記憶システム。

【請求項4】 請求項1に記載の記憶システムにおいて、

前記複数の上位側接続論理装置には、異なるインターフェースを有する前記上位側接続論理装置が含まれることを特徴とする記憶システム。

【請求項5】 請求項1に記載の記憶システムにおいて、

前記記憶装置は複数の小型記憶装置を有し、  
横方向に配列された前記小型記憶装置はECCグループを構成し、  
前記ECCグループが縦方向に配列されることを特徴とする記憶システム。

【請求項6】 上位装置に対するインターフェースを構成する多重化された複数の上位側接続論理装置と、  
上位装置から転送される情報を記憶する記憶装置と、  
前記記憶装置に対するインターフェースを構成する多重化された複数の記憶装置側接続論理装置と、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置との間で転送されるデータを一時記憶する二重化されたキャッシュメモリ装置と、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置とに接続され、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置との間のデータ転送を行う1組のバスとして動作する2系統の高速1/Oバスを含む共通バスと、  
から構成される記憶システム。

【請求項7】 請求項6に記載の記憶システムにおいて、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置はモジュールで構成され、前記モジュールの各々は、それぞれ前記コモンバスに対し挿抜自在に取り付けられることを特徴とする記憶システム。

【請求項8】 請求項6に記載の記憶システムにおいて、  
前記コモンバスはブラッタ上に配設され、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記ブラッタに対し挿抜自在に取り付けられることを特徴とする記憶システム。

【請求項9】 請求項6に記載の記憶システムにおいて、  
前記記憶装置は複数の小型記憶装置を有し、横方向に配列された前記小型記憶装置はECCグループを構成し、  
前記ECCグループが縦方向に配列されることを特徴とする記憶システム。

【請求項10】 請求項6に記載の記憶システムにおいて、  
前記上位装置は2つ以上の前記複数の上位側接続論理装置を介してコモンバスに接続されていることを特徴とする記憶システム。

【請求項11】 請求項6に記載の記憶システムにおいて、  
前記記憶装置は2つ以上の前記複数の記憶装置側接続論理装置を介してコモンバスに接続されていることを特徴とする記憶システム。

【請求項12】 請求項6に記載の記憶システムにおいて、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置は2重化されたマイクロプロセッサと前記プロセッサの動作比較チェックを行うチェック部を有し、  
前記チェック部は前記プロセッサの動作比較チェックを行うことを特徴とする記憶システム。

【請求項13】 請求項6に記載の記憶システムにおいて、  
前記2系統の高速1/Oバスのうち1系統に障害が発生した場合、正常な他の1系統により動作を継続することを特徴とする記憶システム。

【請求項14】 請求項6に記載の記憶システムにおいて、  
前記コモンバスは、さらに前記データ転送のための制御情報を転送する1系統のマルチプロセッサバスを有し、  
前記マルチプロセッサバスに障害が発生した場合、前記2系統の高速1/Oバスのうち一方をマルチプロセッサ

バスとして使用し、他方を高速1/Oバスとして使用することを特徴とする記憶システム。

【請求項15】 上位装置に対するインタフェースを構成し、独立して動作する複数の上位側接続論理装置であって、他の上位側接続論理装置の障害を検出した場合、当該上位側接続論理装置の処理を継続して行うように構成されたプロセッサを有する上位側接続論理装置と、  
上位装置から転送される情報を記憶する記憶装置と、  
前記記憶装置に関するインタフェースを構成し、独立して動作する複数の記憶装置側接続論理装置であって、他の記憶装置側接続論理装置の障害を検出した場合、当該記憶装置側接続論理装置の処理を継続して行うよう構成されたプロセッサを有する記憶装置側接続論理装置と、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置との間で転送されるデータを一時記憶するキャッシュメモリ装置と、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置とを相互に接続する1組のバスとして動作する2系統のデータ転送用バスを含むコモンバスと、  
から構成される記憶システム。

【請求項16】 請求項15に記載の記憶システムにおいて、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置はモジュールで構成され、前記モジュールの各々は、それぞれ前記コモンバスに対し挿抜自在に取り付けられることを特徴とする記憶システム。

【請求項17】 請求項15に記載の記憶システムにおいて、  
前記コモンバスはブラッタ上に配設され、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記ブラッタに対し挿抜自在に取り付けられることを特徴とする記憶システム。

【請求項18】 請求項15に記載の記憶システムにおいて、  
前記記憶装置は複数の小型記憶装置を有し、横方向に配列された前記小型記憶装置はECCグループを構成し、  
前記ECCグループが縦方向に配列されることを特徴とする記憶システム。

【請求項19】 請求項15に記載の記憶システムにおいて、  
前記複数の上位側接続論理装置と前記複数の記憶装置側接続論理装置と前記キャッシュメモリ装置と前記記憶装置とが接続される電源バスと、  
該電源バスに電力を供給する多重化した電源部と、  
前記多重化した電源部に電力を供給する2重化された電源供給部とを有し、



特開平7-20994

該電源部は前記複数の上位側接続論理装置と前記複数の  
記憶装置側接続論理装置と前記キャッシュメモリ装置と

前記記憶装置に必要な数より少なくとも1つ多く接続さ  
れることを特徴とする記憶システム。

B/A  
3

⑤ 日本国特許庁 (J P)

⑥ 特許出願公表

⑦ 公表特許公報 (A)

平5-502525

⑧ 公表 平成5年(1993)4月28日

⑨ Int. Cl. <sup>1</sup>	識別記号	庁内整理番号	審査請求	未請求
G 06 F 15/16	3 7 0 N	9190-5L	予備審査請求	有
13/00	3 5 7 Z	7368-5B		

部門 (区分) 6 (3)

(全 30 頁)

⑩ 発明の名称 並列入出力ネットワーク・ファイル・サーバ・アーキテクチャ

⑪ 特 願 平2-512988

⑫ 翻訳文提出日 平4(1992)3月9日

⑬ 出 願 平2(1990)8月20日

⑭ 国際出願 PCT/US90/04711

⑮ 国際公開番号 WO91/03788

⑯ 国際公開日 平3(1991)3月21日

優先権主張 ⑰ 1989年9月8日 ⑱ 米国 (U S) ⑲ 404,959

⑳ 発 明 者 ロウ エドワード ジョン アメリカ合衆国 カリフォルニア州 94084 マウンテン ヴィン  
マウンテン ローレル コート 468

㉑ 出 願 人 オースベックス システムズ アメリカ合衆国 カリフォルニア州 95054 サンタ クララ パ  
インコーポレイテッド ンカー ヒル レーン 2952

㉒ 代 理 人 弁理士 中村 稔 外6名

㉓ 指 定 国 AT (広域特許), AU, BE (広域特許), CA, CH (広域特許), DE (広域特許), DK (広域特許), ES (広域  
特許), FR (広域特許), GB (広域特許), IT (広域特許), JP, KR, LU (広域特許), NL (広域特許), S  
E (広域特許)

最終頁に続く

請求の範囲

1. データネットワークおよび大容量記憶装置と共に使用されるネットワークサーバ装置であって、

前記ネットワークおよび前記大容量記憶装置と接続可能なインタフェースプロセッサユニットと、

前記ネットワークの顧客ノードによって定義された通信手段を実行できる能力を有するホストプロセッサユニットと

前記インタフェースプロセッサユニットに於ける手段であって、前記大容量記憶装置で前記ネットワークからのデータを記憶するための、前記ネットワークよりのリクエストを満足し、前記大容量記憶装置から前記ネットワークへのデータの検索をするための、前記ネットワークよりのリクエストを満足し、前記ホストプロセッサユニットの処理のため前記ネットワークより前記ホストプロセッサユニットへあらかじめ定義されたカテゴリーのメッセージを送信する手段を含む、

前記送信されたメッセージが前記ネットワークサーバ装置の顧客によって定義された手順を実行するためネットワークの顧客による全てのリクエストを含むことを特徴とするネットワークサーバ装置、

2. 前記インタフェースプロセッサユニットが、

前記ネットワークに接続可能なネットワークコントロールユニットと、

前記大容量記憶装置に接続可能なデータコントロールユニットと、

バッファメモリと、

前記大容量記憶装置で前記ネットワークからの指定記憶データを記憶するため、

前記ネットワークから前記データコントロールユニットへリクエストを送信し、

前記指定記憶データを前記ネットワークから前記バッファメモリへ、そして、

前記バッファメモリから前記データコントロールユニットへ送信し

前記大容量記憶装置から前記ネットワークへの指定検索データを検索するため、

前記ネットワークから前記データコントロールユニットへリクエストを送信し、

前記指定検索データを前記データコントロールユニットから前記バッファメモリへ、そして、前記バッファメモリより前記ネットワークへ送信し、

前記あらかじめ定義されたカテゴリーのメッセージを、前記ホストプロセッサユニットにより処理するため、前記ネットワークから前記ホストプロセッサユニットへ送信する手段を含むことを特徴とする請求の範囲第1項に記載の装置、

3. 前記データコントロールユニットが、

前記大容量記憶装置に接続可能な記憶プロセッサユニットと、

ファイルプロセッサユニットと、

前記ファイルプロセッサユニットでの手段で、

前記ネットワークからの前記ファイルシステムレベル記憶リクエストを前記大容量記憶装置の指定された物理的記憶位置にデータを記憶するためのリクエストに翻訳し、

前記バッファメモリからのデータを、前記大容量記憶装置の前記指定物理的記憶位置へ書き込むように、前記記憶プロセッサユニットに命令し、

前記ネットワークからのファイルシステムレベル検索リクエストを前記大容量記憶装置の前記指定物理的記憶位置からデータを検索するためのリクエストに翻訳し、

もし、前記指定物理的記憶位置から前記データがまだ前記バッファメモリにないならば、データを前記大容量記憶装置の前記指定物理的記憶位置から前記バッファメモリへ検索するように、前記記憶プロセッサユニットに命令する手段および、

前記記憶プロセッサユニットにて、前記バッファメモリと前記大容量記憶装置の間で、データを送信する手段を含むことを特徴とする請求の範囲第2項に記載の装置、

4. データネットワークおよび大容量記憶装置と共に使用されるネットワークサーバ装置であって、

前記ネットワークに接続可能なネットワークコントロールユニットと、

前記大容量記憶装置に接続可能なデータコントロールユニットと、

バッファメモリと、

前記大容量記憶装置で前記ネットワークからの指定記憶データを記憶するための、前記ネットワークからのリクエストを、前記ネットワークコントロールユニ

特表平5-502525 (2)

ットから前記データコントロールユニットへ、送信し、  
前記指定記憶データを前記ネットワークコントロールユニットから前記パフォーマーメモリへDMAにより、そして、前記パフォーマーメモリから前記データコントロールユニットへDMAにより送信し、  
前記指定検索データを前記大容量記憶装置から前記ネットワークへ検索するための、前記ネットワークからのリクエストを、前記ネットワークコントロールユニットから前記データコントロールユニットへ送信し、  
前記指定検索データを前記データコントロールユニットから前記パフォーマーメモリへDMAにより、そして、前記パフォーマーメモリから前記ネットワークコントロールユニットへDMAにより送信する手段をふくむことを特徴とする前記ネットワークサーバー装置、  
5. データネットワーク、大容量記憶装置およびパフォーマーメモリと共に使用され、前記データネットワークからのファイルシステムレベル記憶および検索リクエストに応答するデータコントロールユニットであって、  
前記大容量記憶装置と接続可能な記憶プロセッサユニットと、  
ファイルプロセッサユニットと、  
前記ファイルシステムレベルリクエストを、前記大容量記憶装置の指定物理記憶位置にデータを記憶するためのリクエストに翻訳し、  
前記パフォーマーメモリからデータを前記大容量記憶装置の前記指定物理記憶位置へ書き込むように、前記記憶プロセッサユニットに命令し、  
前記ファイルシステムレベル検索リクエストを、前記大容量記憶装置の前記指定物理記憶位置よりデータを検索するリクエストに翻訳し、  
もし、前記指定物理的位置からの前記データがまだ前記パフォーマーメモリにないならば、データを前記大容量記憶装置の前記指定物理的位置から前記パフォーマーメモリへ検索するように、前記記憶プロセッサユニットに命令する手段および、  
前記記憶プロセッサユニットにて、前記パフォーマーメモリと前記大容量記憶装置の間で、データを送信する手段を含むことを特徴とする前記データコントロールユニット。

ネットワークよりのリクエストを満足し、前記大容量記憶装置から前記ネットワークへのデータの検索をするための、前記ネットワークよりのリクエストを満足し、前記ホストプロセッサユニットで処理のため前記ネットワークより前記ホストプロセッサユニットへあらかじめ定義されたカテゴリのメッセージを送信する手段とを有し、  
前記手段は、  
前記大容量記憶装置で記憶されるため、前記ネットワークからのファイルデータを、前記ネットワークコントロールユニットから前記システムメモリーバンクへ、直接メモリアクセスによって送信し、  
前記大容量記憶装置で記憶されるため、前記ネットワークからのファイルデータを、前記システムメモリーバンクから前記データコントロールユニットへ、直接メモリアクセスによって送信し、  
前記大容量記憶装置から前記ネットワークへの検索のためのファイルデータを、前記データコントロールユニットから前記システムメモリーバンクへ、直接メモリアクセスによって送信し、  
前記大容量記憶装置から前記ネットワークへの検索のためのファイルデータを、前記システムメモリーバンクより前記ネットワークコントロールユニットへ送信する手段を有し、  
少なくとも、前記ネットワークコントロールユニットは、マイクロプロセッサおよび前記システムパフォーマーメモリとは区別されたローカル命令記憶手段を含み、前記マイクロプロセッサの命令の全ては前記ローカル命令記憶手段に常駐していることを特徴とする前記ネットワークノード、  
8. データネットワークおよび大容量記憶装置と共に使用されるネットワークファイルサーバーであって、  
ユニックスオペレーティングシステムで実行されるホストプロセッサユニットと、  
前記ネットワークおよび前記大容量記憶装置と接続可能なインタフェースプロセッサユニットとを含み、  
前記インタフェースプロセッサユニットが、

6. データネットワークおよび大容量記憶装置と共に使用され、前記データネットワークからのファイルシステムレベル記憶および検索リクエストに応答するデータコントロールユニットであって、  
データベースと、  
前記バスに接続されているパフォーマーメモリーバンクと、  
前記バスに接続されていて、前記大容量記憶装置に接続可能な記憶プロセッサ装置と、  
前記バスに接続されていて、ローカルメモリーバンクを有するファイルプロセッサ装置と、  
前記ファイルプロセッサユニットでの手段で、前記ファイルシステムレベルリクエストを、前記大容量記憶装置の指定物理記憶位置にデータを記憶するためのリクエストに翻訳し、前記ファイルシステムレベル検索リクエストを、前記大容量記憶装置の前記指定物理記憶位置よりデータを検索するリクエストに翻訳する手段を含み、  
前記手段は、前記ファイルプロセッサユニットのローカルメモリーバンクによってファイルコントロール情報をキャッシュする手段を含み、  
前記パフォーマーメモリーバンクにより、記憶および検索リクエストに従って、記憶および検索されるように、ファイルデータキャッシュする手段を含むことを特徴とする前記データコントロールユニット、  
7. データネットワークおよび大容量記憶装置と共に使用されるネットワークノードであって、  
システムパフォーマーメモリと、  
前記システムパフォーマーメモリに対して直接メモリアクセスを有するホストプロセッサユニットと、  
前記ネットワークに接続可能で、前記システムパフォーマーメモリにたいして直接メモリアクセスを有するネットワークコントロールユニットと、  
前記大容量記憶装置に接続可能で、前記システムパフォーマーメモリに対して直接メモリアクセスを有するデータコントロールユニットと、  
前記大容量記憶装置で前記ネットワークからのデータを記憶するための、前記

前記ネットワークからの金NFSリクエストを復号し、前記NFSリクエストを満足するための金手順を実行し、前記ネットワークのリターン送信のため、NFS応答のメッセージを符号化し、前記ホストプロセッサユニットで処理のため、前記ネットワークより前記ホストプロセッサユニットに、あらかじめ定義された非NFSカテゴリのメッセージを送信するための手段を有することとを特徴とする前記ネットワークファイルサーバー。

明 細 書

並列入出力ネットワーク・ファイル・サーバ・アーキテクチャ

本出願は、次の米国特許出願に関連しており、これらの出願はすべて同時に出版されている。

- 1. 多機読オペレーションシステム・アーキテクチャ、発明者：ダビッド・ハイフ、アレン・シュヴァルツ、ジェイムス・ロー、ガイ・ハリス。
  - 2. 強化VMEバス・プロトコル使用の同期同期のハンドシェイク及びブロック・モードデータ転送、発明者：ダール・スター。
  - 3. バスロックン・先入れ先出しマルチプロセッサ通信システム使用の同期同期のハンドシェイク及びブロック・モード・データ転送、発明者：ダール・スター、ウィリアム・ビット、ステファン・ブライトマン。
- 上記出願は、すべて、本発明の譲受人に譲渡され、また、すべて本明細書に引用して記載されている。

発明の背景

発明の分野

本発明は、コンピュータ・データ・ネットワークに関し、具体的には、コンピュータ・ネットワークのネットワークファイル・サーバ・アーキテクチャに関する。

関連技術の説明

過去10年の間に、ハードウェアの価格/性能の比率が顕著に増大して、技術とオフィス計算業務の環境に、驚くほどの変化が発生した。分散したワークステーション・サーバ・ネットワークは、大型コンピュータまたはミニコンピュータに付属した、これまでに普及した低速の端末装置を置き換えている。しかし、現在まで、ネットワークの入出力制限により、ワークステーションのユーザが利用出来る潜在的性能は制約されている。マイクロプロセッサの性能の劇的飛躍が、ネットワーク入出力の性能の向上を上回ったために、この状態が一部に発生した。コンピュータネットワークでは、個々のユーザのワークステーションは、クライアントと呼ばれる、ファイリング、プリンティング、データ記憶、広域通信は、サーバと呼ばれる。クライアントとサーバは、すべて、ネットワークの端点と見なされる。クライアント端点は、標準の通信プロトコルを使用して、

子以上に向上した。同時に、これらの非常に高速なクライアントも、遅遅サーバが応えられないデータを求めた。入出力の不足は、ユニックスの場合、非常に劇的であったので、本発明の好適な実施例の説明は、ユニックスのファイルサーバに焦点を合せる。しかし、ユニックスのサーバ入出力問題を解決するアーキテクチャは、他のオペレーションシステムの環境におけるサーバ性能のボトルネックにも容易に拡大する。同時に、好適実施例の説明は、この問題はほかのタイプのネットワークへ容易に拡大するが、イーサネットの実例にも焦点を合せるであろう。

大抵のユニックスの環境において、クライアントとサーバは、サン・マイクロシステム社により普及したユニックス社により広く採用された標準であるRFSを使用して、ファイルデータを交換する。RFSは、"NFS: ネットワーク・システム・プロトコル仕様"、コメント要求(RFC) 1094、サン・マイクロシステム社(1988年3月)の題名のドキュメントに定義されている。このドキュメントは、全体が本明細書に引用されている。

簡単に情報出来るが、RFSは最速ではない。RFSを使用しているクライアントは、ネットワークと、RFSデータをクライアントへ提供するBFSサーバとのいずれにも、かなりの要領を置いている。この要領は、ローカルディスクを所有せず従って2道法アプリケーションと仮想メモリのページ転送及びデータに與与するファイルサーバに依存している、いわゆるディスクのないクライアントに対しては、特に大きい。このユニックスのクライアントサーバの構成では、クライアント出力の10対1の率の増加は、イーサネットの容量、ディスク速度、または、サーバディスクからのネットワーク入出力スループットの10対1の率の増加と釣り合わなかった。

この結果、単一の最近の高域サーバが適切に実現出来る。ディスクなしのクライアントの数は、クライアントの出力とアプリケーションの仕事量に従って、5〜10に低下した。アプリケーションの小規模ローカルディスクとページ転送を有するクライアントの場合、クライアント対サーバの比は、この約2倍か、あるいは、10〜20の間である。

このような低いクライアント/サーバ比率により、ネットワークの構成は、そ

サービス要求を度えて、サーバ端点に依存する。

現在のネットワーク・クライアントとサーバは、一般に、ディスク・オペレーション・システム(DOS)、マッキントッシュ・オペレーション・システム(OS)、OS/2、あるいは、ユニックス・オペレーション・システムを実行する。ローカルネットワークは、一般に、高域ではイーサネットまたはトークリングであり、中域ではアークネット、あるいは、低域ではローカルトークまたはスターLANである。クライアントサーバの通信プロトコルは、オペレーションシステムの環境、すなわち、一般に、パーソナルコンピュータの所有権主権の体系(ネットワーク・3プラス、ワイン、LANマネージャ、LANサーバ)、マッキントッシュのアップルトーク、ユニックスのNFS(ネットワーク・ファイル・システム)またはRFS(リクエスト・ファイル・システム)のTCP(伝送制御プロトコル)/IP(インターネット・プロトコル)の一つにより、非常に厳密に規定されている。これらのプロトコルは、すべて、この業界では広く知られている。

ユニックスのクライアント端点は、一般に、1〜8メガバイトの主メモリ、840×1024画素ディスプレイ、及び、組込みネットワークインタフェースを備えた16または32ビットのマイクロプロセッサであることが特徴である。40〜100メガバイトのローカルディスクは、時にはオプションである。低域の事例は、80286ベースのパーソナルコンピュータまたは88000ベースのマッキントッシュのI型であり、中域領域には、80386パーソナルコンピュータ、マッキントッシュII型、88000ベースのユニックスワークステーションがあり、高域領域には、RISC(既定命令セットコンピュータ)ベースのDEC社、HP社、及びサンユニックス社のワークステーションがある。サーバは、通常では、再構築されたクライアントと全く同じであり、デスク・サイドボックスよりは18インチのラックで構成されている。18インチラックの余分なスペースは、ほかのバックプレーンの際、ディスクまたはテープの駆動装置、及び電源に使用される。

RISCとCISC(複雑命令セットコンピュータ)のマイクロコンピュータの開発により、クライアント・ステーションの性能は、過去の数年間に、10倍

それぞれの部分で、各ローカル・イーサネットが、それ自身の4〜10(ディスクなし)クライアントと専用サーバとの独立した通信量を有するようになる。全体の連結性に関しては、これらのローカルネットワークは、一般に、イーサネットの中核ネットワークと一体に連結するか、あるいは、将来において、FDDI(ファイバ接続デジタルインタフェース)の中核ネットワークと連結する。これらの中核ネットワークは、通常、ローカルネットワークを直接に一体に接続するIP(情報提供ルータ)またはMAC(メディア・アクセス制御)レベルのブリッジによるか、あるいは、ローカルネットワークのすべてについてサーバを接続する、ネットワーク・インタフェースとして機能するもう一つのサーバにより、ローカルネットワークへ接続する。

性能について考慮すべき問題のほかに、低いクライアント対サーバの比率によって、計算問題がいくつかの点で起る。すなわち、

- 1. 共有 5〜10名以上の関係グループは、同じサーバを共有することは出来ず、従って、ファイルのコピーとマニュアルがなく、多くのサーバを更新せず、ファイルを容易に共有することは出来ない。ブリッジとルータは、部分的解決策であるが、さらに多くのネットワークのホップにより、性能上のペナルティを受ける。
  - 2. 資源 システム管理者は、少数より確実なサーバというより、多くの既定容量のサーバを維持しなければならぬ。この責務には、ネットワーク管理、ハードウェア維持、ユーザ教育がある。
  - 3. ファイルシステムの支援 システム管理者またはオペレータは、多くのファイルシステムの支援を行わなければならない。これは面倒で時間のかかる仕事である。各サーバの支援周辺装置を復写転送することも、費用がかかる(あるいは、より速いネットワークの支援装置が使用される場合、各少数のサーバに対し)。
  - 4. 台当りの価格 サーバ当りのクライアントが5〜10程度の場合、サーバのコストは、少数のユーザだけで分割されなければならない。従って、入力レベルのユニックス・ワークステーションの実際のコストは、ワークステーションだけのコストよりも、かなり高く、時には140%も高い。
- 拡大している入出力のギャップと、管理上と経済上の考慮すべき問題は、より

特表平5-502525 (4)

高性能で、より大容量のユニックスファイルサーバの必要性を提示している。ディスクアレイなしのワークステーションをサーバへ転換することは、ディスク容量の問題を扱っているが、基本的入出力の制限を回避することには、何も取っていない。NFSサーバとして、ワルタイム・ワークステーションは、クライアントとしてサポートするように設計されたよりも、5~10倍あるいはそれ以上に、ネットワーク、ディスク、バックプレーン、ファイルスループットを支持しなければならない。より大容量のディスク、より多くのネットワーク・アダプター、予備の主メモリ、あるいは、より高速の処理速度を追加しても、基礎的アーキテクチャの入出力の制約を解決することにはならない。入出力スループットは、十分に増加しない。

特にファイルサーバとして設計されていない、ほかの従来技術のコンピュータアーキテクチャは、それなりに可能性として使用することが出来る。このような周知のアーキテクチャでは、CPU、記憶装置、2台の入出力プロセッサが、単一バスへ接続している。入出力プロセッサの1台は、1組のディスク駆動装置を動作し、アーキテクチャサーバとして使用されるものであれば、ほかの入出力プロセッサがネットワークへ接続する。しかし、少なくとも、2台の入出力プロセッサは、CPUを備えることなくネットワークファイルの要求を処理することが出来ないため、このアーキテクチャは、ファイルサーバとして最適ではない。ネットワーク入出力プロセッサにより受信された、すべてのネットワークファイル要求は、最初にCPUへ送られ、CPUは、このネットワーク要求を受け入れるために、ディスク入出力プロセッサに適切な要求にする。

ほかのこの種のコンピュータアーキテクチャでは、ディスク制御CPUは、ディスク駆動装置へのアクセスを管理し、数台のほかのCPU、例えば、3台のCPUは、ディスク制御CPUの周辺に集められる。ほかの各CPUは、それ自身のネットワークへ接続する。ネットワークCPUは、それぞれ、ディスク制御CPUと、プロセッサ間連絡のために相互に接続する。このコンピュータアーキテクチャの不利な点の一つは、システム内の各CPUがそれぞれ自身の完全なオペレーティングシステムを実行することである。従って、ネットワークファイルサーバの要求は、多数のほかの非ファイルサーバのタスクを行うための機能と処理

過程とにより、重い負荷も受けるオペレーティングシステムによって処理されなければならない。さらに、プロセッサ間連絡は、ファイルサーバのような種類の要求に対して、最大限に利用されない。

さらにはほかのコンピュータアーキテクチャでは、データと命令を格納するそれ自身のキャッシュメモリをそれぞれ備えている、複数のCPUが、システムメモリとディスク制御装置とを有する共通バスへ接続している。このディスク制御装置と各CPUとは、システムメモリへの直接メモリアクセスを有しており、1台以上のCPUは、一つのネットワークへ接続することが出来る。このアーキテクチャは、とりわけ、CPUのファイルデータと命令が、どちらも、同じシステムメモリに存在するので、ファイルサーバとして不利である。従って、CPUが、システムメモリとネットワークCPUとの間を転送される大型ブロックのファイルデータを持つ間、CPUが実行を停止しなければならない場合があるであろう。そのほかに、前述のコンピュータアーキテクチャのいずれかにより、金オペレーティングシステムは、ネットワークCPUより成る各CPUを稼働する。

さらにはほかのタイプのコンピュータアーキテクチャでは、多数のCPUは、相互に接続して一体に接続している。これらのCPUの1台は、ネットワークに接続し、ほかのCPUは、ディスク駆動装置へ接続することが出来る。また、このアーキテクチャは、なかでも、各プロセッサは金オペレーティングを稼働するので、ファイルサーバとしても不利である。プロセッサ間連絡も、ファイルサーバ・アプリケーションに対して最適でない。

発明の要約

本発明は、ユニックス・ファイルサーバの最も共通な動作、すなわち、ファイルオペレーションに対して最大限に利用される。新しいサーバ用特定の入出力アーキテクチャに関する。概要を説明すると、本発明は、1台以上のネットワーク制御装置、1台以上のファイル制御装置、1台以上の記憶プロセッサ、及びシステムまたはバッファメモリを備えているファイルサーバ・アーキテクチャに関するもので、これらの装置は、すべて、メッセージ通過バスに接続し、ユニックス・ホストプロセッサと並行して動作する。ネットワーク制御装置は、それぞれ、1つ以上のネットワークへ接続し、また、サーバ内の他のプロセッサへクワイア

ント要求を連絡するための、ネットワーク層データフォーマットと内部ファイルサーバフォーマットとの間を処理するすべてのプロトコルを運ぶ。ネットワーク制御装置により転送することが出来ないこれらのデータパケット、例えば、クライアント定義のプログラムをサーバで稼働するためのクライアント要求だけは、処理のために、ユニックス・ホストへ送られる。従って、ネットワーク制御装置、ファイル制御装置、及び記憶プロセッサは、金オペレーティングシステムの小部分だけを有し、それぞれは、専用とされる特定タイプの仕事に対し最大限に利用される。

ファイル操作に対するクライアント要求は、記憶プロセッサへ接続した大容量記憶装置の仮想ファイルシステムを、ユニックスホストから独立して、管理するファイル制御装置の一つへ送られる。ファイル制御装置は、また、記憶プロセッサとネットワーク制御装置との間のデータ緩衝をシステムメモリにより制御する。ファイル制御装置は、それぞれ、ファイルデータを一時的格納するシステムメモリから独立して、ファイル制御情報を一時的格納するローカル緩衝メモリを有することが好ましい。さらに、ネットワーク制御装置、ファイルプロセッサ及び記憶プロセッサは、システムメモリからのすべての命令取出しを防止するように、すべて設計されており、その代り、すべての命令メモリを分離した場所に維持する。この構成により、マイクロプロセッサの命令取出しと、メッセージとファイルデータの伝送との間のバックプレーン上の競合は解消する。

図面の簡単な説明

- 図1は、従来技術のファイルサーバ・アーキテクチャの構成図である。
- 図2は、本発明によるファイルサーバ・アーキテクチャの構成図である。
- 図3は、図2に示されたネットワーク制御装置の一つの構成図である。
- 図4は、図2に示されたファイル制御装置の一つの構成図である。
- 図5は、図2に示された記憶プロセッサの一つの構成図である。
- 図6は、図2に示されたシステム・メモリカードの一つの構成図である。
- 図7A~Cは、高速転送プロトコル・ブロック読み取りサイクルの動作を示す流れ図である。

図8A~Cは、高速転送プロトコル・ブロック読み取りサイクルの動作を示す流れ図である。

詳細な説明

目的と背景の比較のために、最初に、事例の従来技術のファイルサーバ・アーキテクチャを図1に開いて説明する。図1は、従来技術によるイーサネットネットワークの通常のユニックススペース・ファイルサーバの全体的構成図である。それは、シングルボードのマイクロプロセッサのホストCPUカード10から成っている。ホストCPUカード10は、イーサネット#1へ接続し、また記憶装置管理装置(MMU)11を経て大型メモリ配列18へ接続している。また、ホストCPUカード10は、キーボード、ビデオディスプレイ、及び2個のRS232C(図示せず)を駆動する。カード10は、MMU11と標準32ビットVMEバス20とを経て、各種の周辺装置へ接続しており、これらの周辺装置には、1台以上のディスク駆動装置24を制御するSMDディスク制御装置22、SCSIバス28へ接続したSCSIホストアダプタ26、1/2インチテープ駆動装置22へ接続したテープ制御装置30と、あるいは、第2イーサネット36へ接続したネットワーク#2の制御装置34などがある。SMDディスク制御装置22は、ディスク制御装置またはバスマスターとして働くMMUにより、バス20とMMU11を經由して直接メモリアクセスして、メモリ配列18と通信することが出来る。この構成は事例としてあり、多くの変形を使用することが出来る。

本システムは、業界標準のTCP/IPとNFSプロトコルスタックを使用して、イーサネットと通信する。プロトコルスタックの一般的説明は、タネンバウムの「コンピュータネットワーク」(第2版、プレントイス・ホール、1988年)に見られる。ファイルサーバ・プロトコルスタックは、525~546ページに記載されている。タネンバウムの参考文献は、本明細書に引用されている。基本的に、次のプロトコル層は、図1の装置に記載されている。

**ネットワーク層** ネットワーク層は、データパケットを、イーサネットに固有のフォーマットと使用される特定タイプのネットワークから独立しているフォーマットとの間で、変換する。図1の装置に使用されているイーサネット固有のフォーマットは、ホーニヒの「イーサネット・ネットワークのIPデータグラム

特表平5-502525 (6)

の転送に関する標準”、RFC 894 (1984年4月)に記載されており、これは本明細書に引用されている。

インターネット・プロトコル (IP) 層 この層は、相互接続したネットワークシステムの発信元から送信元へ、ビットパッケージ (インターネット・データグラム) を送るのに必要な機能を提供している。ファイルサーバからクライアントへ送られるメッセージに関して、サーバの高レベル層は、IPモジュールを呼出して、転送する送信元クライアントのインターネットアドレスとメッセージとを送る。IPモジュールは、すべての介在するゲートウェイのパケットサイズの制限に適合するように、必要なすべての、メッセージの分割を行い、各断片にインターネット・ヘッダーを加え、ネットワーク層に生成したインターネット・データグラムを送ることを要求する。インターネット・ヘッダーには、ローカルネットワーク送信元アドレス (インターネット・アドレスから翻訳された) とはかのパラメータがある。

ネットワーク層からIP層より受信されたメッセージに関して、IPモジュールは、データグラムは、ほかのネットワーク、例えば、図1の35などの第2イーサネットのほかのホストへ送らなければならないか、または、サーバ自身宛のものであることを、インターネット・アドレスから決定する。第2ネットワークのほかのホスト宛であるならば、IPモジュールは、送信元のローカルネットワークに要求する。データグラムがサーバ内のアプリケーションプログラム宛であるならば、IP層はヘッダーを削除し、メッセージの残りの部分を対応する次の高い層へ通過させる。図1の事例の領域に使用されているインターネットプロトコル標準は、インフォメーション・サイエンス・インスティテュートの“インターネットプロトコル、DARPAインターネット・プログラム・プロトコル仕様”、RFC 791 (1981年9月)に記載されており、本明細書に引用されている。

TCP/UDP層 この層は、IP層よりさらに詳細な実装とアドレス指定のオプション付のデータグラムサービスである。例えば、IPデータグラムは、約1,500バイトを保持し、ホストへアドレス指定することが出来るが、UDPデ

ータグラムは、約4キロバイトを保持し、ホスト内の特定のポートへアドレス指定することが出来る。TCPとUDPは、この層において既製のプロトコルである。取付けられた信頼出来る、データの流れを送ることを必要とするアプリケーションは、TCPを使用し、これに対し、取付けられ信頼出来る送りを必要としないアプリケーション (NFSなど) はUDPを使用する。

図1の従来技術によるファイルサーバは、TCPとUDPのどちらにも使用する。すなわち、ファイルサーバは、ファイルサーバ関連のサービスにUDPを使用し、サーバがネットワークのクライアントへ提供されるいくつかのサービスにTCPを使用する。UDPは、ポストエルの“ユーザ・データグラム・プロトコル”、RFC 768 (1980年9月28日)に記載されており、本明細書に引用されている。TCPは、ポストエルの“伝送制御プロトコル”、RFC 761 (1980年1月)とRFC 793 (1981年9月)に記載されており、これも本明細書に引用されている。

XDR/RPC層 この層は、指定された手続きを遠隔の機械で実行するために高レベルプログラムから呼び出し可能な機能を提供している。また、この層は、クライアントの機械が手続きをサーバに実行させるに必要な解決を行う。例えば、クライアントの拠点における呼び出し者のプロセスは、呼び出しメッセージを図1のサーバへ送る。呼び出しメッセージは、所望の機械の仕様とそのパラメータから成っている。メッセージは、このスタックを越えてRPC層へ送られ、この層はサーバ内の対応する手続きを呼び出す。手続きが完了すると、回答メッセージが生成して、RPCはそれをスタックへ伝え、ネットワークを越えて呼び出し者クライアントへ戻す。RPCは、サン・マイクロシステム社の“RPC、遠隔手続き呼び出しプロトコル仕様、第2版”、RFC 1075 (1988年6月)に記載されており、本明細書に引用されている。

RPCは、基本的UDP層を出入して通過した情報を表すXDR (外部データ指示標準) を使用している。XDRは、単にデータ符号化標準であり、異なるコンピュータアーキテクチャ間でデータを送るために有用である。このようにして、XDR/RPC層のネットワーク側では、情報は、機械から独立している。ホストアプリケーション側では、そうではない。XDRは、サン・マイクロシ

ステム社の“XDR、外部データ指示標準” RFC 1114 (1987年6月)に記載されており、本明細書に引用されている。

NFS層 NFS (ネットワーク・ファイル・システム) 層は、RPC要求が呼び出し出来るサーバに使用出来るプログラムの一つである。ホストアドレス、プログラム番号、及びRPC要求内の手続き番号の組合せは、呼出されるべき、一つの通常のNFS手続きを指定することが出来る。

遠隔手続きがNFSを図1のファイルサーバに呼出すことにより、ディスク24の分割されたファイルへの、即座で無所蔵の遠隔検出が行われる。NFSは、全体として登録簿による高レベルのファイルにより、階層的であるファイルシステムを仮定する。クライアントホストは、各種手続きから成る約20NFS手続きのすべてを呼出すことが出来る。これらの手続きには、指定された数のバイトを指定されたファイルからの読取り、指定された数のバイトを指定されたファイルへの書込み、指定されたファイルの作成、名前変更、除去、登録簿木の構文解析、登録簿の作成と除去、及び、ファイル属性の設定などがある。データが格納されたまま換装される範囲内のディスク上の位置は、ファイル処理または転送指定とバイトオフセットなどにより、論理項に常に指定される。実際のデータ記憶の詳細は、クライアントからは隠されている。NFS手続きは、ユニックスのVFSとUFSなどの可能な高レベルモジュールと共に、駆動装置、磁気ヘッド、トラック、セクター識別などの物理的データアドレスへの論理データアドレスの転換をすべて行う。NFSは、サン・マイクロシステム社の“NFS、ネットワーク・ファイルシステム・プロトコル仕様”、RFC 1094 (1988年8月)に記載されており、本明細書に引用されている。

ネットワーク層の可能性のある例外を除いて、上述のプロトコル処理は、すべて、ホストCPUカード10内の単一プロセッサにより、ソフトウェアで行われる。すなわち、イーサネットパケットがイーサネット12に到着すると、ホストCPU10は、NFSスタック内のプロトコル処理と、ホスト10で実行しているほかのアプリケーションのプロトコル処理とを、すべて進行する。データとプログラムコードに関してMMU11を頼りて行われているメモリ18へのアクセスにより、NFS手続きは、ホストCPUで実行される。論理的に指

定されたデータアドレスは、さらに物理的に指定されたフォームへ変換され、VMEバス20を経てSMDディスク制御装置またはSCS1ホストアダプタ28へ伝えられる。また、ホストCPUは、VMEバス20を経てテープ制御装置30と通信して、ファイルサーバの各種の機械的動作を行う手続きを実行する。これらのなかで、クライアントのワークステーションにより要求されたクライアント定義の遠隔手続きがある。

サーバが第2イーサネット36を動作するならば、そのイーサネットからのパケットは、同じVMEバス20を越えてIPデータグラムの形で、ホストCPU10へ送られる。その上、ネットワーク層を除くプロトコル処理は、すべて、ホストCPU10で実行するソフトウェア処理過程により進行される。さらに、イーサネット12または36のどちらかのサーバから送られるすべてのメッセージのプロトコル処理は、また、ホストCPU10で実行している処理過程により行われる。

次に、2台の各イーサネットに、5~10のクライアントが、高い頻度で、ファイルサーバへの要求を行い、また、回答を受け取ることを必要とするならば、ホストCPU10は、莫大な量のデータ処理を行うことは、明らかである。ホストCPU10は、多重タスク・ユニックス・オペレーションシステムを実行する。従って、入信する各要求は、前の要求が完全に処理されるか、処理される前に戻されるのを待つ必要はない。多重処理過程は、各種要求の処理を異なる段階で行うため、ホストCPU10で行われ、従って、多くの要求が、同時に、処理過程にある。しかし、カード10には1台のCPUしかないため、これらの要求の処理は、本当の意味の並行方式で達成されるものではない。従って、CPU10は、主要なボトルネックをファイルサーバ要求の処理に示している。

ほかのボトルネックがMMU11に発生する。MMU11は、CPUカード10とメモリ18の間で、命令とデータの両方を転送する。ディスク駆動装置とネットワークの間を流れるデータは、すべて、このインタフェースを2倍以上通過する。

さらにはほかのボトルネックがVMEバス20に発生する。バス20は、SMDディスク制御装置22、SCS1ホストアダプタ28、ホストCPUカード10、

特表平5-502525 (6)

及び、あるいはネットワーク制御装置24の間でデータを転送する。

好適実施例一全ハードウェア・アーキテクチャ

図2には、本発明によるネットワーク・ファイルサーバ100の構成図が示されている。サーバ100は、多重ネットワーク制御装置(NC)ボード、1枚以上のファイル制御装置(FC)ボード、1枚以上の記憶プロセッサ(SP)ボード、多重システムメモリボード、及び、1台以上のホストプロセッサから成っている。図2に示す特定の実施例は、4枚のネットワーク制御装置ボード110a~110d、2枚のファイル制御装置ボード112a~112b、2枚の記憶プロセッサ114a~114b、合計で192メガバイト・メモリの4枚のシステムメモリ・カード118a~118d、及び1台のローカルホストプロセッサ118を有する。ボード110、112、114、118、及び118は、VMEバス120で一体に接続しており、VMEバス20には、前に確立した強化VMEバス・プロトコル・アプリケーションで説明した強化ブロック転送モードが使用されている。図2に示す4台のネットワーク制御装置110は、それぞれ、合計容量が8台のイーサネット122a~122hとして、2台のイーサネットまで接続することが出来る。記憶プロセッサ114は、それぞれ、10ラインの並列SCSIバスを動作し、そのうちの9ラインは、それぞれ、3台のSCSIディスク駆動装置を支援することが出来る。各記憶プロセッサ114の10番目のSCSIチャンネルは、テープ駆動装置とは別のSCSI周辺装置に使用される。ホスト118は、本質的に、標準のサン・オープン・ネットワーク・コンピュータインジ(ONC)サービスを提供する。重要なことは、ユーザ定義の手続きを実施する、すべてのネットワーク要求が、実行のためにホストへ送られることである。NCボード110、FCボード112、SPボード114は、それぞれ、それ自身の独立した32ビットマイクロプロセッサを有する。これらのボードは、ホストプロセッサ118から、実際には、NFSとIP処理のすべてから、本質的に負荷を受けていない。イーサネット122のクライアントに出入するメッセージの大部分は、NFSの要求と回答から成っているため、NC、FC、SPのプロセッサによるこれら要求の並行処理は、ローカルホスト

118による掛かり合いを最小にして、ファイルサーバの性能を非常に大きく改良するものである。ユニックスは、実質的にすべてのネットワーク、ファイル、記憶処理から明らかに取り除かれている。

全ソフトウェア構成とデータの流れ

図2のハードウェアのサブシステムを詳細に考察する前に、ソフトウェアの構造の概要を説明する。ソフトウェア構成は、上述のアプリケーション、名称“多重性能オペレーティングシステム・アーキテクチャ”で、詳しく説明されている。

ソフトウェアの要素の多くは、業界では周知であり、大抵の構成されたユニックスシステムで見られるが、まだ知られていない二つの構成要素がある。すなわち、ローカルNFS(“LNFS”)とメッセージング中核(“MK”)オペレーティングシステム中核である。最初、これら二つの構成要素を説明する。

メッセージング中核

ファイルサーバ100の各種プロセッサは、各プロセッサ110、112、114、118を稼動するメッセージング中核を使用して、相互に通信する。これらのプロセッサは、命令メモリを共有しないので、タスクレベル通信は、従来のユニックスで発生する以上に、莫大に手書き呼出しによって発生しない。その代り、メッセージング中核は、メッセージをVMEバス120に通過させて、必要なプロセッサ相互間の通信をすべて行う。メッセージ通信は、単純性と速度の理由で、通常手書き呼出しより好適である。

メッセージング中核により通過したメッセージは、固定の128バイトの長さがある。単一プロセッサ内で、メッセージが参照により渡される。プロセッサ間で、メッセージは、メッセージング中核により複製され、次に、宛先処理過程へ参照によって送られる。図2のプロセッサは、以降に説明するように、都合よくプロセッサ間のメッセージング中核メッセージ交換して稼働する。

LNFS-ローカルNFSインタフェース

2機能NFS標準は、信頼性のない通信を使用する無所異のオペレーション内に特に設計された。これは、クライアントもサーバも、防しているとき互いに聞いていないければ、確信出来ないことを意味する(不確信性)。実際に、イーサネット環境では、これは良好に動作する。

しかし、サーバ100のなかでは、NFSレベルデータグラムは、プロセッサ

間、特にネットワーク制御装置110とファイル制御装置112の間、及びホストプロセッサ118とファイル制御装置112の間の通信にも使用される。殆どよく便利であるが、この内部通信に関して、完全な無所異または信頼出来ない通信を有することは、望ましくなくまた実行不可能である。従って、既知NFS、すなわち、LNFSは、NFS要求と回答との内部通信に使用される。LNFSは、ファイルサーバ100のなかでのみ使用される。このサーバにより支援された外部ネットワークプロトコルは、貴重な標準の、認可されたNFSである。LNFSは、以降に詳しく説明する。

ネットワーク制御装置110は、すべてのプロトコル処理がNFSまで行われた後に外部NFS要求と回答、及び内部LNFS要求と回答との間で変換するNFSサーバを、それぞれ、稼動する。例えば、NFS要求は、XDR付でUDPデータグラムに内封されたRPC要求として到着する。プロトコル処理の後、NFSサーバは、NFS要求をLNFSの形に翻訳し、メッセージング中核を使用して要求をファイル制御装置112へ送る。

ファイル制御装置は、ネットワーク制御装置とホスト118とからのLNFS要求を知照するLNFSサーバを稼動する。LNFSサーバは、LNFS要求をファイルシステム・サーバに対応した形へ翻訳し、また、ファイル制御装置を稼動する。この制御装置は、ブロック入出力層により、システムメモリファイルデータキャッシュを管理する。

各プロセッサのソフトウェアの概要を以下に説明する。

ネットワーク制御装置110

サーバ100の最大に活用されたデータの流は、知所ネットワーク制御装置110から始まる。このプロセッサは、イーサネットパケットをクライアントのワークステーションから受信する。プロセッサは、NFS宛先のパケットを製造し、次に、パケットへの全プロトコル処理をNFSレベルに行い、生成したLNFS要求を直接にファイル制御装置112へ送る。このプロトコル処理には、IPの経路指定と再組立、UDPのデマルチプレキシング、XDRの暗号解読、及び、NFSの要求タスク指名がある。この逆の設備は、NFS回答をクライアントへ送り返すために、使用される。重要なことは、これらの時間を消費す

る作業が、ホスト118でなく、ネットワーク制御装置110で直接に行われる。サーバ100は、サン・マイクロシステム社、マウンテンビュー、カルフォルニア州、から導入した通常のNFSを使用しており、これは、互換性のあるNFSプロトコルである。

LNFSネットワーク通信量が、その宛先ホストプロセッサ118へ直接に送られる。

ネットワーク制御装置110は、自身のIP経路指定も行う。各ネットワーク制御装置110は、2台の完全並列のイーサネットを支援する。サーバ100の実施例のネットワーク制御装置が図2に示されており、従って、サーバは、8台のイーサネットまで支援することが出来る。

同一ネットワーク制御装置に2台のイーサネットの場合、IP経路指定が、ネットワーク制御装置内で完全に発生し、バックプレーンのトラフィック量は発生しない。従って、同一制御装置へ2台の相互に他動的イーサネットを稼動すると、そのイーサネット通過時間を最小にするだけでなく、また、VMEバス120のバックプレイン融合をかなり促進する。経路指定テーブル更新は、ホストプロセッサ118からネットワーク制御装置へ送られ、これはゲート付または経路指定されたユニックス・デモンを実行する。

ここで説明したネットワーク制御装置は、イーサネットLAN内に設計されているが、本発明は、FDDIを含めて、ほかのネットワークタイプにもまったく容易に使用出来ることは、理解されるであろう。

ファイル制御装置112

別個のプロセッサをNFSプロトコル処理とIP経路指定に専用化するほかに、サーバ100も、別個のプロセッサ、知所ファイル制御装置112を、すべてのファイルシステム処理に対し責任を持つように専用化する。そのプロセッサは、従来のバークレイ・ユニックス4.3ファイルシステムコードを使用し、ディスク上の2進互換性データ表示を使用している。これらの二つを選択することにより、すべての標準ファイルシステム・ユーティリティ(特に、ブロックレベル・ツール)が稼動することが出来る。

ファイル制御装置112は、すべてのネットワーク制御装置110とホスト

## 特表平5-502525 (7)

ロセッサ118により使用された共有ファイルシステムを稼動する。ネットワーク制御装置とホストプロセッサ114は、LNFSインタフェースを使用しているファイル制御装置112と通信する。ネットワーク制御装置110は上記のようにLNFSを使用し、ホストプロセッサ118は、サン・オーエスの標準仮想ファイルシステム (VFS) インタフェースへのプラグイン・モジュールとしてLNFSを使用する。

ネットワーク制御装置がNFS読み取り要求をクライアントのワークステーションから受信すると、その結果のLNFS要求がファイル制御装置112へ通過する。ファイル制御装置112は、最初に、要求されたデータのシステムメモリ118バッファ・キャッシュを探索する。見つけ出すと、バッファへの参照がネットワーク制御装置110へ戻される。見つけ出さないと、システムメモリ118内の最低使用頻度 (LRU) キャッシュバッファは、解放されて、要求されたブロックへ再割り当てられる。次に、ファイル制御装置は、記憶プロセッサ114に、ブロックをディスク駆動部からキャッシュバッファへ読み取ることを命令する。記憶プロセッサ114はこれをファイル制御装置へ通知し、ファイル制御装置は、順次、ネットワーク制御装置110へ通知する。次に、ネットワーク制御装置110は、バッファからのデータと一緒に、NFS回答をネットワークのNFSクライアントワークステーションへ送り戻す。留意すべきことは、記憶プロセッサ114が、このデータをシステムメモリ118へ送り、必要ならば、ネットワーク制御装置110が、システムメモリ118からのデータをネットワークへ送ることである。この処理過程は、ホスト118を含めることなく、行われる。

### 記憶プロセッサ

知的記憶プロセッサ114は、すべてのディスクとテープの記憶動作を管理する。自律的であるが、記憶プロセッサは、第一に、ファイル制御装置112により指示されて、ファイルデータをシステムメモリ118とディスク・サブシステムの間を移動する。ホスト118とファイル制御装置112のいずれも実際のデータ経路から除外することによって、多くの遠隔クライアントにサービスに必要な性能を速くすることに役に立つ。

かのサーバ機能に使用され、二つのプロセッサの間の信頼出来るバイトストリーム通信を行う。ソケットは、TCP接続を設定するために使用される。

**VFSインタフェース** 仮想ファイルシステム (VFS) インタフェースは、標準的サンオーエスファイルシステム・インタフェースである。これは、ユーザとユニックス・オペレーティングシステムの非ファイル部分とに於ける一様なファイルシステム・ピクチャで置き、特定のファイルシステムの詳細を隠す。従って、標準NFS、LNFS、及びローカルユニックス・ファイルシステムは、調和して共存する。

**UFSインタフェース** ユニックス・ファイルシステム (UFS) インタフェースは、ローカルプロセッサ・ディスク駆動装置と通信するための、従来タイプで熟知のユニックス・インタフェースである。サーバ100では、このインタフェースは、ファイル制御装置を通過することなく、ときどき記憶プロセッサ・ボリュームを直接に取り付けるために使用される。一般に、ホスト118はLNFSを使用し、ファイル制御装置を通過する。

**記憶層** 記憶層は、ユニックス標準モデルと異なる物理的装置の実装と間の標準ソフトウェアインタフェースである。サーバ100では、ディスク装置は、ホストプロセッサへ直接に取り付けられていないので、ホストの記憶層内のディスク駆動装置は、メッセージング中継を使用して、記憶プロセッサ114と通信する。

**ルートとポートのマッピング・デモン** ルートとポートのマッピング・デモンは、パケット経路指定のルートとポートのデータベースを維持する、ユニックス・ユーザレベル背景プロセスである。このプロセスは、非常に不活性であり、すべての実行経路には存在しない。

**イエローページとオーセンチケータ・デモン** イエローページとオーセンチケータ・サービスは、サンONC標準ネットワークサービスである。イエローページは、多目的の、名前付き人名簿検索サービスである。オーセンチケータ・サービスは、要求を権限する、すなわち検索するための暗号キーを使用して、要求者が、すべての行動または希望するデータに関し、正当な権限を有することを保証する。

そのほかに、ホスト118のサーバ管理プログラムにより調整されて、記憶プロセッサ114は、ディスク・サブシステムと、テープあるいはSCSIチャンネルのほかに文書保管周辺装置との間でデータを移動することにより、サーバを支援する。さらに、ホストプロセッサ118によって直接にアクセスされるならば、記憶プロセッサ114は、非常に高性能な、ユニックス用の従来タイプのディスクインタフェース、仮想メモリ、データベースを提供することが出来る。ホストプロセッサ118は、ブート、記憶交換、生分割を記憶プロセッサ114により取り付けることが出来る。

各記憶プロセッサ114は、10個の並列な、完全同期SCSIチャンネル (バス) を同時に作動する。これらチャンネルのうち1個は、9個のSCSIディスク駆動部を8列をそれぞれ支援し、一つの列にある各駆動部は、異なるSCSIチャンネルへ割り当てられる。10番目のSCSIチャンネルは、7個のテープとは別のSCSI周辺装置までホストとして対応する。読み取りと書き込みを行うほかに、記憶プロセッサ114は、ディスク探索時に行先判定などの装置レベルの最大利用を行い、装置エラー回復を命令し、また、装置とシステム118の間のDMA転送を制御する。

### ホストプロセッサ118

ローカルホスト118は、三つの主要な目的を有する。すなわち、ユニックスの稼動、クライアントに対する標準ONCネットワークサービスの提供、及び、サーバ管理プログラムの実行である。ユニックスとONCは、標準サンオーエス・リリース4とONCサービス・リリース2とから取り入れているので、サーバ100は、イエローページ、ロックマネージャ、デーイーエス・オーセンチケータ、オート・マウンタ、及びポート・マッピングなどの交換性のある、高レベルONCサービスを同時に行うことが出来る。テルネット、FTP、SMTP、SNMP、及び逆ARPなどのサン/2ネットワークディスク・パーティングとより一般的IPインターネット・サービスも支援される。最後に、プリントサーバと両様なユニックスデモンは、即応的に作動する。

ホストプロセッサ118は、次のソフトウェア・モジュールを稼動する。

**TCPとソケット層** 転送制御プロトコル (TCP) は、NFS以外のいくつ

**サーバ管理プログラム** サーバ管理プログラムは、構成、論理エラー、性能レポートを制御する管理アプリケーションの一式のプログラムであって、モニタリングとチューニングのインタフェースをシステム管理者へ提供する。これらの機能は、ホスト118へ接続したシステムコンソール、あるいは、システム管理者のワークステーションから行われる。

ホストプロセッサ118は、普通のOEMサン中央プロセッサカード、モデル3E/120である。これは、モトローラ68020マイクロプロセッサと4メガバイトの記憶ボードを組み込んでいる。SPARCベースのプロセッサなどのほかのプロセッサも、使用可能である。

サーバ100のハードウェアの各構成要素の構造と動作を以降に説明する。

### ネットワーク制御装置のハードウェア・アーキテクチャ

図3は、ネットワーク制御装置110aの実例のデータ経路といくつかの制御経路を示す構成図である。これは、32ビットマイクロプロセッサ・データバス212へ接続した20MHz 68020マイクロプロセッサ210を備えている。また、256キロバイトCPUメモリ214は、マイクロプロセッサ・データバス212へ接続している。マイクロプロセッサ・データバス212の低位の8ビットは、双方向性バッファ218を経て、8ビット低速データバス218へ接続している。低速データバス218には、128キロバイト読み可能なプログラムROM (EPROM) 220、32バイトプログラムROM (PROM) 222、及び多機能周辺装置 (MEP) 224が接続している。EPROM220は、ネットワーク制御装置110aのファームコードを有し、PROMは、ボード上の2台のイーサネット・インタフェースのそれぞれへ割り当てられたイーサネット・アドレスなどの各種オペレーティング・パラメータを格納している。イーサネットアドレス情報は、初期値設定の際に、CPUメモリ214の対応するインタフェース制御ブロックへ読み込まれる。MPP224は、モトローラ68901であって、タイピング、読み込み、及び汎用入出力などの各種ローカル機能を行う。また、MPP224は、RS232ポート228へインタフェースするための方針制御受送信機 (UART) を有する。これらの機能は本発明には重要でなく、さらに詳しく説明しない。



特表平5-502525 (8)

マイクロプロセッサ・データバス212の低位の18ビットも、双方向性バッファ230を経て16ビットLANデータバス232へ接続している。LAN制御装置チップ234、例えば、Am7990 LANCEイーサネット制御装置、アドバンスド・マイクロ・デバイス社、サニーベイル、カルフォルニア製造、はLANデータバス232を図2の第1イーサネット122aにインタフェースさせている。LAN制御装置234の制御とデータは、512キロボイトLANメモリ236に格納され、メモリ236もLANデータバス232へ接続している。特殊な16〜32ビット先入れ先出し(FIFO)チップ240は、FIFOパリティチップと呼ばれ、以降に説明するが、これもLANデータバス232へ接続している。また、LAN直接メモリアクセス(DMA)制御装置242も、LANデータバス232へ接続しており、装置242は、LANメモリ236とFIFOチップ240との間のデータパケットの移動を制御する。LAN-DMA制御装置242は、チャンネルゼロだけを使用しているモトローラM68440 DMA制御装置であってよい。

図2に示す第2イーサネット122bは、図3に示すネットワーク制御装置カード110aの第2LANデータバス252へ接続している。LANデータバス252は、双方向性バッファ250を経て、マイクロプロセッサ・データバス212の低位18ビットへ接続しており、LANデータバス232に見られるものと同様な構成要素を有している。具体的には、LAN制御装置254は、データと制御のLANメモリ256を使用して、LANデータバス252をイーサネット122bとインタフェースさせており、また、LAN-DMA制御装置252は、LANメモリ256とパリティFIFO260の16ビット幅のデータポートAとの間のデータのDMA転送を制御する。

マイクロプロセッサ・データバス212の低位18ビットは、もう一つのパリティFIFO270へも直接に接続し、また、VME/FIFO DMA制御装置272の制御ポートへも接続している。FIFO270は、以降に説明するように、CPUメモリ214と、遠隔ボード110、112、114、116、または、118(図2)の一つとの間でメッセージを送るために使用される。VME/FIFO DMA制御装置は、データコピー用の三つのラウンドロビン方式の

排他先チャンネルを支援しており、遠隔ボードの一つとFIFO240、260、または270のいずれかとの間、及び、FIFO240と260との間のすべてのデータ転送を制御する。

32ビットデータバス274は、FIFO240、260及び270のそれぞれの32ビットのポートBへ接続しており、これらの転送が行われるデータバスである。データバス274は、双方向性逐次制御方式ラッチ278を経由して、ローカル32ビットバス276と通信し、次に、ラッチ278は、双方向性バッファ280を経由して、VMEバス120と通信する。

また、ローカルデータバス276は、1組の制御レジスタ282に接続しており、レジスタ282はVMEバス120を通過して直接にアドレス指定可能である。レジスタ282は、主にシステムの初期値設定と移行に使用される。

ローカルデータバス276は、また、双方向性バッファ284を経て、マイクロプロセッサデータバス212へ接続している。ネットワーク制御装置110aがスレーブモードで動作すると、VMEバス120から直接にアドレス指定可能である。遠隔ボードの一つは、双方向性バッファ284を経て、CPUメモリから直接にデータを復写することが出来る。LANメモリ236と260は、VMEバス120を通過して直接にアドレス指定されない。

パリティFIFO240、260、及び270は、それぞれ、特定用途向IC(ASIC)から成っており、その機能と動作は、付属書に記載されている。FIFO240と260は、パケットデータ転送に関して構成され、FIFO270は、メッセージ通過に関して構成されている。付属書に照して、FIFO240と260は、データ転送構成レジスタの次のビット設定により、プログラムされる。

ビット	定 義	設 定
0	WDモード	N/A
1	パリティチップ	N/A
2	パリティ正モード	N/A
3	3/18ビットCPUとポートA インタフェース18ビット	(1)
4	反転ポートAアドレス0	00 (0)
5	反転ポートAアドレス1	YES (1)
6	検査合計折返しけた上げ	YES (1)
7	リセット	00 (0)

データ転送制御レジスタは、次のようにプログラムされる。

ビット	定 義	設 定
0	使用可能ポートA要求/応答	YES (1)
1	使用可能ポートB要求/応答	YES (1)
2	データ転送命令	待望通り
3	CPUパリティ使用可能	00 (0)
4	ポートAパリティ使用可能	00 (0)
5	ポートBパリティ使用可能	00 (0)
6	検査合計 使用可能	YES (1)
7	ポートA マスタ	YES (1)

FIFO240と260に使用した構成と異なり、マイクロプロセッサ210は、ポートAを直接にローディングとアンローディングする実態を有する。マイクロプロセッサ210は、単一の命令により、2サイクルのポートAアクセスを使用して、金32ビットワードをポートAから読み取る。ポートAのデータ転送は、データ転送制御レジスタの0(使用可能ポートA要求/応答)と7(ポートAマスタ)を設定かつ外すことにより、不能になる。

FIFO270の制御設定の残りは、上述のFIFO240と260の残りと同じである。

ネットワーク制御装置110aは、また、コマンドFIFO280を有する。

コマンドFIFO280は、ローカルデータバス276へ接続した入力ポートを有しており、また、VMEバス120を通過して直接にアドレス指定可能であって、マイクロプロセッサデータバス212へ接続した出力ポートを有する。以降に厚詳細に説明するように、遠隔ボードの一つが、コマンドまたは回答をネットワーク制御装置110aへ送り出す場合、ボードは、1ワード(32ビット)メッセージ記述子を制御装置110aのコマンドFIFO280へ直接に書き込むことにより、コマンドを送る。コマンドFIFO280は、“FIFO280空いていない”状態をマイクロプロセッサへ発信し、次に、マイクロプロセッサ210は、このメッセージ記述子をFIFO280のトップから読み取り、これを格納する。メッセージがコマンドであれば、メッセージには、メッセージが位置しているVMEアドレスがある(多分、遠隔ボードの一つにある214と同様な共有メモリ内のアドレス)。次に、マイクロプロセッサ210は、FIFO270とVME/FIFO DMA制御装置272とをプログラムして、メッセージを遠隔位置からCPUメモリ214へ復写する。

コマンドFIFO280は、普通の2ポートFIFOであるが、FIFOが満たされている間にデータ入力ポートへ書き込もうとする場合、ほかの回路が、バス・エラー番号をVMEバス120に発生するために備えられていることを除く。コマンドFIFO280は、256入力のためのスペースを有する。

ネットワーク制御装置110aのアーキテクチャの顕著な特徴は、LANバス232と252がマイクロプロセッサデータバス212から独立していることである。イーサネットへ送信されそこから発信されるデータパケットは、LANデータバス232のLANメモリ236に格納され(あるいは、LANデータバス252の258に)、CPUメモリ214には格納されない。LANメモリ236、258とイーサネット122a、122bとの間のデータ転送は、LAN制御装置234と254によりそれぞれ制御され、LANメモリ236または256とVMEバス120の遠隔ポートとの間の大部分のデータ転送は、LAN-DMA制御装置242と262、FIFO240と260、及びVME/FIFO DMA制御装置272により制御される。このルールの例外が、データ転送のサイズが小さい場合、例えば8バイトより小さい場合に発生する。この場合、マ

### 特表平5-502525 (9)

マイクロプロセッサ210は、DMAを使用せずに、それを直接に復写する。マイクロプロセッサ210は、転送を介して完了時に通知を受けることを除いて、より大きい転送には関与しない。

CPUメモリ214は、マイクロプロセッサ210への命令、FIFO270を経て送達ボードへ送られるまたそこから送り出されるメッセージ、及び、FIFO DMA制御装置、LAN制御装置を制御するための各種データブロックを大部分構成している。プロトコル処理のために、双方向性バッファ230と250により直接アドレス指定することによって、マイクロプロセッサ210は、データパケットをLANメモリ236と256にアクセスする。従って、CPUメモリ214内のローカル高速静止RAMは、ゼロ待ち状態のメモリアクセスを、ネットワーク通信量から独立したマイクロプロセッサ210へ提供することが出来る。これは、図1の従来技術のアーキテクチャと明確な対照をなしており、従来技術の場合、データとデータパケット、及びホストCPUカード10向けのマイクロプロセッサ命令は、メモリ18に存在しており、MMU11を介してホストCPUカード10と通信しなければならない。

LANデータバス232と252は、図3に分岐したバスとして示されているが、それらのバスは、代わりに、単一の組合せられたバスとして実施出来ることは、理解されるであろう。

#### ネットワーク制御装置の動作

動作中に、LAN制御装置の一つ(例えば234)が、情報のパケットをそのイーサネット122aから受信すると、制御装置は全体のパケットに読み取り、それを対応するLANメモリ236に格納する。次に、LAN制御装置234は、読み取りをMPP224を介してマイクロプロセッサ210へ渡し、マイクロプロセッサ210は、LAN制御装置234の状態レジスタを調べ(双方向性バッファ230をより)、読み取りが発生した事象が、“完了した受信パケット”であったことを決定する。MPP224の優先読み取り特性によって発生した第2イーサネット122bの潜在的ロックアウトを防止するため、マイクロプロセッサ210は、この時点では、受信したパケットを処理しない。その代り、このような処理は、ポーリング・ファンクションにスケジューされる。

ポーリング・ファンクションが、受信したパケットに到達すると、パケットに対する制御は、ソフトウェア・リンクレベル受信モジュールへ渡される。次に、リンクレベル受信モジュールは、二つの異なるフレームフォーマットのいずれかに従って、パケットを復号する。すなわち、標準イーサネットフォーマットまたはSNAP(IEEE802LCC)フォーマットである。パケットのヘッダの入口は、どのフォーマットが使用されたかを記入する。次に、リンクレベル受信モジュールは、3種類のメッセージのどれが、受信したパケットに含まれているかを決定する。すなわち、その種類は、(1)IP、(2)ローカルARPモジュールにより処理出来るARPパケット、または(3)処理するためにローカルホスト118(図2)へ送らなければならないARPパケットとほかのタイプのパケット。パケットが、ネットワーク制御装置110aにより処理出来るARPパケット例えば、サーバ110のアドレスの要求などであれば、マイクロプロセッサ210は、回答パケットをLANメモリ236にアセンブルし、これにより、従来の方法で、LAN制御装置234はそのパケットをイーサネット122aへ送り戻すことが出来る。注目すべきことは、CPUメモリ214の指示により制御されて、マイクロプロセッサによって直接にアドレス指定され、このタスクを達成するためのデータ処理が、LANメモリ236ではほぼ完全に実行されることである。このファンクションは、VMEバックプレーンに通信量を少しも発生することなく、また、ローカルホスト118を混乱することもなく、達成される。受信されたパケットが、ネットワーク制御装置110aで完全に処理されないARPパケットであるか、あるいは、ローカルホスト118への通信を必要とするほかのタイプのパケット(例えば、サーバ110がクライアント定義の手続きを実行することに対するクライアント要求)であるならば、マイクロプロセッサ210は、LAN DMA制御装置242がパケットをLANメモリ236からFIFO240へロードするようにプログラムし、FIFO240がデータ転送の方向にプログラムし、また、DMA制御装置272がFIFO240からVMEバス120を渡ってシステムメモリ118へ読み込むようにプログラムする。具体的には、マイクロプロセッサ210は、最初に、LAN DMA制御装置242にLANメモリ236内のパケットの実行開始アドレスと長さをプログラムし、次に、

この制御装置が、FIFOがデータ受信の準備が出来ると直ちに、データをLANメモリ236からパリティFIFOのポートAへ転送し始めるようにプログラムする。第2に、マイクロプロセッサ210は、VME/FIFO DMA制御装置272にシステムメモリ118内の宛先アドレスとデータパケットの長さをプログラムして、この制御装置に、FIFO280のポートBからVMEバス120へのデータ転送を始めるように命令する。最後に、マイクロプロセッサ210は、FIFO240に行う転送の方向をプログラムする。次に、転送は、マイクロプロセッサ210のこれ以上の関与もなく、DMA制御装置242と272の全自動的制御のもとで進行する。

次に、マイクロプロセッサ210は、パケットが指定されたシステムメモリ・アドレスで使用可能であるホスト118へ、メッセージを送る。マイクロプロセッサ210は、メッセージ記述子をホスト上のソフトウェアがエミュレートされたコマンドFIFOへ書き込むことにより、このメッセージを送り、ホストは、メッセージを、通常のVMEブロック転送モードで、ネットワーク制御装置のCPUメッセージ214から、バッファ284を経て、ホストのローカルメモリへ復写する。次に、ホストは、通常のVME転送により、パケットをシステムメモリ118からホスト自身のローカルメモリへ復写する。

ネットワーク制御装置110aによりネットワークから受信されたパケットがIPパケットであるならば、マイクロプロセッサ210は、パケットが、(1)NFSパケットでないサーバ110向けのIPパケットか、(2)ほかのネットワークへ送られるべきIPパケットか、あるいは、(3)NFSパケットであるかを決定する。それがサーバ110向けのIPパケットであるか、NFSパケットでないならば、マイクロプロセッサ210は、いくつかのARPパケットに関して上述した同じ方法で、パケットをLANメモリ236からホスト118へ送る。

IPパケットがサーバ110向けとされていないが、ほかのネットワークのクライアントへ送られるべきであるならば、パケットは、宛先クライアントが接続しているイーサネットと連結したLANメモリへ復写される。宛先クライアントがイーサネット122bにあって、イーサネットが宛先イーサネット122aと同じネットワーク制御装置ボード上にあるならば、マイクロプロセッサ210は、

パケットをLANメモリ236からLAN256へ復写し、次に、LAN制御装置254がパケットをイーサネット122bへ転送する(当然のことであるが、二つのLANデータバス232と252が結合しているならば、復写は必要でない。マイクロプロセッサ210により、単に、LAN制御装置254は、パケットがLAN制御装置234により書き込まれたLANメモリ内の同じ場所から、パケットを読み取るだけである)。

LANメモリ236からLANメモリ256へのパケットの復写は、上述のLANメモリからシステムメモリへの復写と同様に実行される。8バイト以上の転送サイズに関して、マイクロプロセッサ210は、最初に、LAN DMA制御装置242に、LANメモリ236のパケットの実行開始アドレスと長さをプログラムし、また、FIFOがデータを受信する用意が出来ると直ちに、制御装置が、LANメモリ236からパリティFIFO240のポートAへデータを転送し始めるようにプログラムする。第2に、マイクロプロセッサ210は、LAN DMA制御装置282に、LANメモリ256の宛先アドレスとデータパケットの長さをプログラムし、さらに、制御装置は、データをパリティFIFO280からLANメモリ256へ転送するように命令する。第3に、マイクロプロセッサ210は、VME/FIFO DMA制御装置272が、データワードをFIFO240のポートBから、データバス274を経て、FIFO280のポートBへ計量するようにプログラムする。最後に、マイクロプロセッサ210は、二つのFIFO240と280に、行われる転送の方向をプログラムする。次に、転送が、マイクロプロセッサの関与もなく、DMA制御装置242、282、272の全自動的制御のもとで進行する。LANメモリからシステムメモリへの復写のように、転送サイズが8バイトより小さいならば、マイクロプロセッサ210は、DMAなしで、転送を直接に行う。

LAN DMA制御装置242と282は、それぞれ、その内容を完了すると、MPP224により形成された各種読み取りにより、そのことはマイクロプロセッサ210に通知する。マイクロプロセッサ210が両方の読み取りを受信すると、マイクロプロセッサ210は、LAN制御装置254が、通常の方式でパケットをイーサネット122bへ送るようにプログラムする。

特表平5-502525 (10)

以上のように、単一ネットワーク制御装置110の2台のイーサネット間の1P転送指定は、データバス274を通して行われ、VMEバス120には送信量を発生しない。また、図1の従来技術のアーキテクチャと対照して、ホストプロセッサ118は、このような転送指定によって選択されもしない。その上、最終の復写動作は、DAM制御装置とパリティPIFOのプログラミングと、それらを開始する命令との重複ファンクションに属してのみ、マイクロプロセッサ210外部の制御装置により行われ、マイクロプロセッサ210と、マイクロプロセッサ・データバス212のバス送信量を必要とする。VME/PIFO DMA制御装置272は、ローディング制御レジスタによりマイクロプロセッサ・データバス212を経てプログラムされる。LAN DMA制御装置242と282は、各制御装置のローディング制御レジスタにより、マイクロプロセッサ・データバス212、各双方向性バッファ230と250、及び各LANデータバス232と250を経て、プログラムされる。パリティPIFO240と260は、付属書に記載されているように、プログラムされる。

送られる1Pパケットの宛先ワークステーションが、イーサネット、ネットワーク制御装置110の一つへ接続する場合、パケットは、イーサネットが接続されているネットワーク制御装置110の対応するLANメモリへ復写される。このような復写は、いくつかのARPパケットに属して上述したように、最初にパケットをシステムメモリ118へ復写することにより達成され、次に、宛先ネットワーク制御装置にパケットが使用出来ることを通知する。ネットワーク制御装置がそのように通知されると、制御装置は、自身のパリティPIFOとDMA制御装置が、パケットをシステムメモリ118から対応するLANメモリへ復写するようにプログラムする。重要なことは、このタイプの1P転送指定がVMEバス送信量を生成しなくても、これは、やはり、ホストCPU118を関与させていない。

イーサネット122aから受信して、現在ではLANメモリ238に格納されている1Pパケットが、サーバ100向けとされているNFSパケットであるならば、マイクロプロセッサ210は、すべての必要なプロトコル処理を行って、NFSメッセージを抽出し、それをローカルNFS (L NFS) フォーマットへ

変換する。これは、LANメッセージ238に格納された、多数の個々の1Pパケットから抽出したデータの論理的連続を構成しており、リンクされたリストをCPUメッセージ214に生成し、異なるブロックのデータをLANメモリ238に正しい順序で格納する。L NFSフォーマットの正確な詳細は、次に記載すべき点を除いて、本発明を理解するには重要でない。すなわち、フォーマットは、記憶プロセッサ114へ取り付けられたディスクに格納されているファイルの登録簿を維持するコマンドと、このディスクのファイルに対しデータを読取り/書き込みを行うコマンドと、各種の構成管理と診断制御のメッセージとを有する。L NFSにより支援された登録簿維持コマンドは、通常のNFSをベースにした次のメッセージを有する。すなわち、ファイルの属性を得る (GETATTR)、ファイルの属性を設定する (SETATTR)、ファイルを開示する (LOOKUP)、ファイルを生産する (CREATE)、ファイルを取り外す (REMOVE)、ファイルを再命名する (RENAME)、新しいリンクしたファイルを生産する (LINK)、シムリンクを生産する (SYMLINK)、登録簿を取り外す (RMDIR)、及び、ファイルシステムの統計量を戻す (STATFS) である。L NFSにより支援されたデータ転送コマンドは、ファイルからの読取り (READ)、ファイルへの書き込み (WRITE)、登録簿からの読取り (READDIR)、及び、リンクの読取り (READLINK) を有する。L NFSは、また、ファイル制御装置に、ネットワーク制御装置が、システムメモリ内の指定されたバッファを使用して終了していることを通知するために、バッファリリースコマンド (RELEASE) を支援する。また、L NFSは、与えられたタイプのアクセスが指定ファイルに指定された資格に対し正当であるか、ないかを決定するために、VOP 導出アクセスを支援する。

L NFS要求が、LANメモリ238からディスクへのファイルデータの書き込みを含んでいるならば、ネットワーク制御装置110aは、最初に、対応するファイル制御装置112により割り当てられるべきシステムメモリ118のバッファを要求する。バッファへのポインタが戻ると、マイクロプロセッサ210は、LAN DMA制御装置242、パリティPIFO240及びVME/PIFO DMA制御装置272が、ファイルデータの全ブロックをシステムメモリ118へ送るようにプログラムする。この転送と、1PパケットとARPパケットをシ

ステムメモリ118への転送に属して上述した転送との唯一の差は、LANメモリ238全体に分散した部分を、一般に有することである。マイクロプロセッサ210は、前記の部分が完了した通知を受けた後、リンクしたリストを使って、LAN DMA制御装置242をデータの各部分に属して連続的にプログラムすることにより、その状態を通知させる。マイクロプロセッサ210は、全データブロックがシステムメモリ118に連続的に転送されなければならない限り、パリティPIFO240とVME/PIFO DMA制御装置272を全メッセージに属して一度プログラムすることが出来る。そうでないならば、マイクロプロセッサ210は、LAN DMA制御装置と同じ方法で、連続的ブロックに属してDMA制御装置272をプログラムすることが出来る。

ネットワーク制御装置110aが、サーバ100のほかのプロセッサから、一般にはファイル制御装置112からメッセージを受信する場合、そのファイルデータは、イーサネットの一つに、例えばイーサネット122aに転送するために、システムメモリ118で使うことが出来る。次に、ネットワーク制御装置110aは、逆方向のファイルデータの復写と同じ方法で、ファイルデータをLANメモリ238へ復写する。具体的には、マイクロプロセッサ210は、最初に、システムメモリ118内の実行開始アドレスと長さにより、VME/PIFO DMA制御装置をプログラムし、次に、PIFOがデータ受信の用意が出来ると直ちに、この制御装置が、データを、VMEバス120を経てパリティPIFO240のポートBへ転送し始めるようにプログラムする。次に、マイクロプロセッサ210は、LAN DMA制御装置242をLANメモリ238内の宛先アドレスとファイルデータの長さによりプログラムし、その制御装置にデータパリティPIFO240からLANメモリ238へ転送するように命令する。第3に、マイクロプロセッサ210は、パリティPIFO240を行われる転送の方向でプログラムする。次に、マイクロプロセッサ210が少しも関与することなく、この転送は、DMA制御装置242と272の全面的制御のもとで進行する。再度説明すると、データファイルがシステムメモリ118内の多数のブロックに分散する場合、マイクロプロセッサ210は、VME/PIFO DMA制御装置を、正しい順序で転送するブロックのリンクしたリストでプログラムする。

DMA制御装置242と272が作業を完了すると、それらの装置は、そのことをMFP224を介してマイクロプロセッサ210へ通知する。次に、マイクロプロセッサ210は、イーサネット1Pパケットの形でイーサネット122aを通過して転送するメッセージを作成するために、LANメモリ238内の L NFSメッセージについて、すべての必要なプロトコル処理を行う。前記のように、このプロトコル処理は、ローカルホスト118が関与することなく、ネットワーク制御装置110aで完全に行われる。

注目すべきことは、パリティPIFOが、多数の128バイトブロックを非常に効率よく移動するように設計されていることである。ポートBを通過するデータ転送サイズは、常に32ビット幅であり、32ビット幅に相当するVMEアドレスは、整列された4倍のバイトでなければならない。バス利用のために、相当するローカル開始アドレスが整列された2倍のバイトである場合、それは16ビットに設定され、さもなければ、8ビットに設定される。TCP/IP検査合計は、常に16ビットモードで演算される。従って、ローカル開始アドレスが整列された2倍のバイトでないならば、検査合計ワードは、バイトの交換を必要とする。

従って、PIFO240、280または270のすべてのポートBからポートAへの転送に属して、マイクロプロセッサ210は、転送数を次の128バイト境界に埋め込むように、VME/PIFO DMA制御装置をプログラムする。余剰の32ビットワード転送は、VMEバスを必要とせず、所要数の32ビットワードだけが、ポートAから取り出される。

パリティPIFO270のポートAからポートBへの転送に属しては、マイクロプロセッサ210は、ポートAにワードがローディングし、それが完了すると、PIFOの完全表示を行う。PIFO完全表示により、ポートBからのアンローディングが可能になる。また、128バイトより少ない転送が、LAN DMA制御装置242または282の制御装置のもとでなく、ローカルプロセッサ制御のもとで行われるので、同じ手続きが、パリティPIFO240または280のポートAからポートBの転送について行われる。PIFOのすべてに属し、VME/PIFO DMA制御装置が、所要の数の32ビットワードだけが

## 特表平5-502525 (11)

取り出すようにプログラムされる。

### ファイル制御装置ハードウェア・アーキテクチャ

ファイル制御装置 (FC) 112 は、それぞれ、モトローラ社などの、既製の標準マイクロプロセッサボードでよい。しかし、図4に構成図の形で示されているような専用ボードが好適である。

図4は、FC112aの一つを示しており、ほかのFCも同じであることは、理解されるであろう。多くの面で、それは、単に図3のネットワーク制御装置 (NC) 110aの縮小型であり、いくつかの点で、拡大されている。NC110aのように、FC112aは、32ビットマイクロプロセッサデータバス312に接続した20MHz 88020マイクロプロセッサ310を有する。また、256キロバイトの共有CPUメモリ314が、マイクロプロセッサ・データバス312へ接続されている。低位8ビットのマイクロプロセッサ・データバス312は、双方向性バッファ316を経て、8ビット低速データバス318へ接続している。低速データバス318には、128キロバイトPROM (プログラマブルROM) と多機能周辺装置 (MFP) 324がある。PROM320とMFP324の機能は、NC110aのEPROM220とMFP224とについて上述した機能と同じである。FC112aは、NC110aのEPROM220のようなEPROMを有していないが、並列ポート392を有する。並列ポート392は、主に試験と診断に使用される。

NC110aのように、FC112aは、双方向性バッファ380と32ビットローカルデータバス378を経て、VMEバス120へ接続している。1組の制御レジスタ382は、ローカルデータバス378へ接続し、VMEバス120を通して直接にアドレス指定可能である。また、ローカルデータバス378は、双方向性バッファ384を経てマイクロプロセッサデータバス312へ接続している。これは、VMEバス120からのCPUメモリ314の直接アドレス指定を可能にしている。

FC112aは、コマンドFIFO390を有しており、FIFO390は、ローカルデータバス378へ接続した入力ポートを有し、また、VMEバス120を通して直接にアドレス指定可能である。コマンドFIFO390は、また、マ

クロプロセッサデータバス312へ接続した出力ポートを有する。コマンドFIFO390の構造、動作、目的は、NC110aのコマンドFIFO290について上述したものと同一である。

FC112aは、NC110aには存在するLANデータバス323と352を省略しているが、その代りに、双方向性バッファ384を経てマイクロプロセッサデータバス312へ接続した4メガバイト・32ビット幅FCメモリ396を有する。以降に明らかになるが、FCメモリ396は、ファイル制御情報のキャッシュメモリとして使用され、システムメモリ118に一時隠されたファイル情報から分離している。図4のファイル制御装置の実施例は、DMA制御装置を備えていない。従って、VMEバス120を通してブロック転送モードでデータを転送または受信するマスタとして働かない。ブロック転送は、CPUメモリ314とFCメモリ396により、しかし、VMEバススレーブとして働くFC112aにより、始動する。このような転送では、通常のマスタは、双方向性バッファ384と、適切ならば、384を経て、VMEバス120を直接に通して、CPUメモリ314またはFCメモリ396をアドレス指定する。

### ファイル制御装置の動作

FC112aの目的は、基本的に、VMEバス120上の遠隔プロセッサによりLNFSフォーマットに形成された要求に応答して、仮想ファイルシステムサービスを提供することである。大部分の要求は、ネットワーク制御装置110から送られるが、また、要求はローカルホスト118からも送られる。

LNFSにより受渡されたファイル関係コマンドは、上記のように識別される。そのコマンドは、理論的に識別されたディスクデータブロックに関して、FC112aへすべて指定される。例えば、ファイルからのデータを読み取る LNFS コマンドは、読み取る先のファイルの指定 (ファイルシステムID (FSID) 及びファイルID (inode))、バイトオフセット、及び読み取るバイト数のカウントを有する。FC112aは、コマンドを満足するため、その識別を物理的形式へ、すなわち、ディスクとセクタ数へ変換する。

FC112aは、普通の高速ファイルシステム (FPSまたはUPS) を駆動し、このシステムは、パークレー4.8 VAXリリーズをベースにしている。しか

し、前述のように、制御データキャッシュ (一時隠し) は、FC112a上のFCメモリ396を使用して行われ、ディスクデータキャッシングは、システムメモリ118 (図2) を使用して行われる。FC112a内のこのファイル制御装置をキャッシングすることにより、ファイル制御装置がシステムメモリ118にキャッシュされる場合発生する、VMEバスの輻射と速度低下とを防止することが出来る。FC112aのメモリは、三つの主目的のために、VMEバス120を通して直接にアクセスされる。キャッシュされたファイル制御装置を読み取るか、または、書き込み記憶プロセッサ114によるFCメモリ396へのアクセスは、非常に頻繁に行われる。これは、局部的に修正されたファイル制御装置をディスクに読み込むか、または、ファイル制御装置をディスクから読み取るようにFC112aにより要求されたアクセスである。第2に、FCのCPUメモリ314が、FC112aからこのほかのプロセッサへのメッセージ転送のために、ほかのプロセッサにより直接にアクセスされる。例えば、システムメモリ内のデータブロックが、ディスクへ書き込むために記憶プロセッサへ転送されなければならない場合、FC112aは、最初に、このような転送を要求するローカルメモリ314内のメッセージをアセンブルする。次に、FC112aは、記憶プロセッサ114へ通知し、プロセッサ114は、CPUメモリ314から直接にメモリを複製し、要求された転送を実行する。

FCローカルメモリへの直接アクセスの第3のタイプは、LNFSクライアントが登録項目を読み取る場合に発生する。FC112aが登録項目を読み取るLNFSの要求を受信すると、FCメモリ396内の要求された登録項目をフォーマットし、その場所の要求者へ通知する。次に、要求者は、項目を読み取るために、FCメモリ396を直接にアクセスする。

FC112aのUPSコードには、二つのキャッシュを分離するためのいくつかの修正がある。具体的には、2組のバッファ・ベクターが、FCメモリ396に一つとシステムメモリに一つ、維持されている。さらに、もう1組のシステム・バッファ・ルーチン (GETBLK(), BRELSR(), BREAD(), BRWITC()) と、BREDA() が、FCメモリ396に一つとシステム118へのバッファアクセスに一つ、存在する。UPSコードは、ファイル制御装置へアクセスするために

FCメモリの対応するバッファルーチンを呼出すため、また、ディスクデータのキャッシングのためにシステムメモリ118の対応するバッファルーチン呼出すために、さらに修正される。UPSの記述は、*・由コンサルティングのリークンとウェブ (サンタクララ、カルフォルニア、1988年)* 著「検索の構造と流れ」の2章、6章、7章、8章に見られ、これは本明細書に引用されている。

読み取りコマンドが、ネットワーク制御装置などの要求者よりFCへ送られると、FCは、最初に、ファイル、オフセット、カウントの情報をディスクとセクター情報へ変換する。次に、FCは、その情報を有するシステムバッファをロックし、記憶プロセッサ114に、必要ならば、それらをディスクから読み取ることを命令する。バッファが準備出来ると、FCは、指定されたファイルの属性と、データを保持するシステムメモリ118の位置を識別するバッファ記述子の配列とを有する要求者へメッセージを戻す。

要求者がデータをバッファから読み取ると、リリーズ要求をFCへ送り戻す。リリーズ要求は、FCにより読み取り要求に回答して戻された同じメッセージと同じである。FC112aは、どのバッファを解放するか決定するために、取納された情報を使用する。

書き込みコマンドは、読み取りコマンドと同様に、FC112aにより処理されるが、呼出し者は、FC112aにより戻されたバッファ記述子により識別されたシステムメモリ118の記憶場所へ (読み取り代りに) 書き込むことを要求される。FC112aは、ライトスルー・キャッシュを使用するので、リリーズコマンドを要求者から受信すると、FC112aは、記憶プロセッサ114に、可能な限り前記のためにシステムメモリバッファを解放する前に、システムメモリ118からのデータを対応するディスクセクターに書き込むことを命令する。

BEADIR トランザクションは、読み取りと書き込みに似ているが、この目的のために特に要求された登録項目をフォーマットした後、要求は、FC112aにより自身のFCメモリ396から直接に満足される。情報がすでに局部的にキャッシュされていなければ、FC112aにより、プロセッサはディスクから要求された登録項目を読み取る事が出来る。また、指定されたオフセットは、バイ

特表平5-502525 (12)

トオフセットの代りの“マジック・キュー”であり、ファイルの絶対バイトオフセットの代りに、登録項目を識別する。

READIR トランザクションも、ファイル属性を張さない。リンクは常にその全体に読み込まれるので、トランザクションは、オフセットまたはカウントを必要としない。

システムメモリ118により行われるディスクデータキャッシュのすべてについて、FC112aは、バッファトラックの動的な、制御、制御解除、及び保持に関する中央許可機関として働く。二つ以上のFC112aがある場合、各FCは、システムメモリ118の自身に割り当てられた部分に排他的制御を行う。これらトランザクションのすべてにおいて、要求されたバッファは、初期の要求と解放の要求との間の期間を越して、ロックされる。これは、ほかのクライアントによるデータの破壊を防止する。

また、二つ以上のFCがある場合、ディスク上の各ファイルシステムは、FCの特定の一つに割り当てられる。FC#0は、FCバイスプレジデントと呼ばれる如き過程を保持し、この過程は、ファイルシステムが割り当てられているFCのリストを維持する。クライアントプロセッサ(例えば、NC110)が、特定のファイルシステムを指定するLNFS要求を行うと、それは最初に、メソッド内のfileをFCバイスプレジデントへ送り、どのFCが指定されたファイルシステムを制御するのか質問する。FCバイスプレジデントは応答し、クライアントプロセッサはLNFS要求を指定されたFCへ送る。クライアントプロセッサは、また、FCバイスプレジデントへのこの要求の数を最小にするように、fileをFCのペアの自身のリストを押し出すと、それらを維持管理する。

記憶プロセッサ・ハードウェア・アーキテクチャ

ファイナリー100において、各記憶プロセッサ114は、VMEバス120を10個までの異なるSCSIバスにインタフェースさせることが出来る。さらに、プロセッサ114は、毎秒55メガバイトの強化ブロック転送プロトコルの最高使用速度においてそのようにすることが出来る。

図5は、記憶プロセッサ114aの一つの構成図である。記憶プロセッサ114bも同様である。記憶プロセッサ114aはマイクロプロセッサ510を備えている。

り、このプロセッサは、20MHzで動作するモトローラ68020マイクロプロセッサでよい。マイクロプロセッサ510は、32ビットマイクロプロセッサデータバス12を経て、CPUメモリ514と接続し、1メガバイトまでの静止RAMを有する。マイクロプロセッサ510は、ほかの発生器との結合もなく、命令、データ及び状態を専用バス12でアクセスする。マイクロプロセッサ510は、バス12の唯一のマスタである。

マイクロプロセッサデータバス12の低位16ビットは、双方向性バッファ518を経て、制御バス518とインタフェースする。制御バス518の低位8ビットは、ほかの双方向性バッファ522を経て、低速バス520とインタフェースしている。低速バス520は、NC110a(図3)のMFP224と同様にMFP524へ接続し、また、NC110aのPROM220と同様にPROM528と接続している。EPROM528の幅と速度により、機能コードは、高速実行のためにリセットされると、CPUメモリ514へ復写される。

MFP524は、NC110aのMFP224のように、モトローラ68901多機能周辺装置を備えている。この装置は、一つの方向性制御済み制御装置、複数の独立のプログラマブル入出力ピン、四つのタイマー、及び一つの非同期受送信機(UART)の機能を提供している。UARTの機能は、デバッグのモニターと診断のために、RS232バス(図示に示されていない)を經由して遠隔通信を行う。四つのタイミング回路のうち二つは、独立の、または、解放状に、汎用タイマーとしてマイクロプロセッサ510により使用される。8番目のタイマーの機能は、下記のようにDMA制御に付し再生制御を行う。MFP524のほかの機能は、モトローラ社の“MC68901多機能周辺装置”に見られ、これは本明細書に引用されている。MFP524が備えている8個の一般目的入出力ビットは、次の表に従って構成されている。

ビット	命令の定義
7入力	電力故障は緊急である-これは早期警告として機能する。
6入力	SCSI 注意-合成SCSI 10の全SCSIチャンネルから注意
5入力	チャンネル動作完了-合成チャンネルはDMA制御装置の13の全チャンネルからのビットを完了、後述。
ビット	命令の定義
4出力	DMA制御装置実行可能。DMA制御装置を起動する。
3入力	VMEバス制御完了-VMEバス制御の完了を表示。
2入力	コマンド使用可能-SPのコマンドPifo(後述)が一つ以上のコマンドポインタを有することを表示。
1出力	外部制御不能。マイクロプロセッサへの外部発生制御不能不能。
0出力	コマンドPifo 実行可能。SPのコマンドPifoの動作を可能。リセットの場合、コマンドPifoをクリア。

コマンドは、双方向性バッファ530、ローカルデータバス532、及びコマンドPIFO534を経て、VMEバス120からSP114aへ送られる。コマンドPIFO534は、NC110aとPC112aのコマンドPIFO290と390とにそれぞれ接続しており、256の32ビット項目の探索を有する。コマンドPIFO534は、VMEバス120に見られる書き込み専用レジスタであり、マイクロプロセッサ510に見られるように読取り専用レジスタである。PIFOが、VMEバスからの書き込みの初めに満たされているならば、VMEバスのエラーが発生する。ポインタは、受信した順序でまたマイクロプロセッサ510によってのみ、コマンドPIFO534から取り出される。コマンド使用可能な状態は、MFP524の入出力ビット4により形成され、一つ以上のコマンドポインタがなおコマンドPIFO534内にある限り、コマンド使用可能な状態は、継続して表示される。

前述のように、SP114aは、10個までのSCSIバスまたはチャンネル

540a~540jを支援する。一般的構成では、バス540a~540iは3つまでのディスク駆動装置をそれぞれ支援し、チャンネル540jは、テープ駆動装置、光学ディスクなどのほかのSCSI周辺装置を支援する。他動的に、SP114aは、超小型Dサブコネクタとツイードケーブルとにより、各SCSIバスに接続している。8本の50ピンケーブルが、バス番号118の番号と12の接地を伝導する300の導体を備えている。ケーブルは、SP114aの前面パネルとディスク駆動装置のコミュニケーションボードとに取付けられている。標準の50ピンケーブルは、各SCSI装置をコミュニケーションボードへ接続する。終端レジスタがSP114aに取り付けられている。

SP114aは、各SCSIバス540上の、毎秒5メガバイトまでの同期並行データ転送、開閉、及び接続解除/再接続の作業を支援する。各SCSIバス540は、各SCSIアダプタ542へ接続し、アダプタは、本実施例では、アダプタ社、ミルビタス、カルフォルニア、の製作によるAIC8250制御装置ICであり、これは、非多重アドレスモードで動作する。SCSIアダプタ542は、それぞれ、その各SCSIチャンネルを実行するための、必要なハードウェアのインタフェースと低レベル電氣的プロトコルとを備えている。

各SCSIアダプタ542の8ビットデータポートは、1組が10個のパリティPIFO544a~544jの各PIFOのポートAへ接続している。PIFO544aは、NC110aのPIFO240、280、270と同じであり、また、各SCSIアダプタ542の8ビットデータポートと36ビット(32ビット+パリティの4ビット)の共通データバス550との間にパリティ適用のデータ転送を行うように、接続して、構成されている。PIFO540は、このために、ハンドシェイク、状態、ワードの組立/分解、及び速度適合のPIFO制御を行う。また、PIFO540は、32ビットパリティを生成して、検査し、また、RA105実行のために、重複したデータを累積して、検査し、回復したデータを累積する。

パリティPIFO544がすべて存在するように、SCSI542アダプタ542は、すべて、マイクロプロセッサ510のアドレススペースの単一位置に存在する。所望の組合せを指すために、最初に1組の選択レジスタ(示されてい

特表平5-502525 (13)

ない)モプログラムし、次に、組合せの所望のチップの制御レジスタアドレスから読み取るか、または、書き込むかにより、マイクロプロセッサ510は、個々の制御レジスタと組合せてアクセスのFIFOを選択する。マイクロプロセッサ510は、制御バス518とほかの双方向性バッファ548を経て、SCS1アダプタ542の制御レジスタと通信し、また、制御バス518と双方向性バッファ552を経て、FIFO544の制御レジスタと通信する。SCS1アダプタ542とFIFO544は、いずれも、8ビット制御レジスタを使用し、FIFO544のレジスタ・アドレス指定は、このようなレジスタが連続したバイト位置に別名で存在するように、構成されている。これにより、マイクロプロセッサ510は、単一の8ビットレジスタとして、そのレジスタに書き込むことが出来、これによって、命令のオーバーヘッドを低減することが出来る。

パリティFIFO544は、それぞれ、アダプタ542の8250方式で構成されている。付属書に関して、FIFO544は、データ転送用レジスタに次のようにビットを設定することにより、プログラムされる。

ビット	定義	設定
0	WDモード	1(0)
1	パリティチップ	(1)
2	パリティ正モード	(0)
3	8/16ビットCPUとポートAインタフェース18ビット	(0)
4	反転ポートAアドレス0	(1)
5	反転ポートAアドレス1	(1)
6	検査合算折返し上げた上げ	(0)
7	リセット	(0)

データ転送制御レジスタは、次のようにプログラムされる。

データ転送制御レジスタは、次のようにプログラムされる。

ビット	定義	設定
0	使用可能ポートA要求/応答	(1)
1	使用可能ポートB要求/応答	(1)
2	データ転送命令	待望通り
3	CPUパリティ使用可能	(0)
4	ポートAパリティ使用可能	(1)
5	ポートBパリティ使用可能	(1)
6	検査合算使用可能	(0)
7	ポートAマスタ	(0)

さらに、RAMアクセス制御レジスタ(ロングバースト)の4ビットは、8バイトバーストに関してプログラムされる。

FIFO544のポートAは、16ビット制御バス518へ接続し、ポートBは、共通データバス550へ接続する。FIFO544は、以降に詳しく説明するように、マイクロプロセッサ510がVMEバス120と直接に通信出来る一つの手段を提供している。

マイクロプロセッサ510は、16チャンネルの1組を使用して、データ移動を管理し、各チャンネルは、現在の状態を示す唯一のステータスを有する。チャンネルは、いずれも制御バス518に接続したチャンネル使用可能レジスタ560とチャンネルステータス・レジスタ562とを使用して、実行される。チャンネル使用可能レジスタ560は、16ビット書き込み専用レジスタであるが、チャンネルステータス・レジスタ562は、読取り専用レジスタである。二つのレジスタが、マイクロプロセッサ510と同じアドレスに存在する。マイクロプロセッサは、各ビットをチャンネル使用可能レジスタ560に設定することにより、特定のチャンネルを使用可能にし、チャンネルステータスレジスタ562の“完了”ビットについて検査することにより、指定された作業の完了を確認する。次に、マイクロプロセッサ510は、使用可能ビットをリセットし、これにより、チャンネルステータスレジスタ562の各“完了”ビットはクリアされる。チャンネルは次のように定義される。

ビット	定義	設定
0	使用可能ポートA要求/応答	(1)
1	使用可能ポートB要求/応答	(1)
2	データ転送命令	待望通り
3	CPUパリティ使用可能	(0)
4	ポートAパリティ使用可能	(1)
5	ポートBパリティ使用可能	(1)
6	検査合算使用可能	(0)
7	ポートAマスタ	(0)

そのほかに、RAMアクセス制御レジスタ(ロングバースト)の4ビットは、8バイトバーストに関してプログラムされる。

SCS1アダプタ542は、それぞれ、各制御信号を発生し、その状態は、18ビットSCS1新込みレジスタ556の10ビットとして、マイクロプロセッサ510へ送られる。SCS1新込みレジスタ558は、制御バス518へ接続する。さらに、複合SCS1新込みは、SCS1アダプタ542のいずれか一つが処理を必要とするときはいつでも、MPP524により形成される。

ほかのパリティFIFO544も、また、メッセージ渡しのために、SP114に接続されている。再復、付属書に関して、パリティFIFO544は、データ転送用レジスタの次のビット設定によりプログラムされる。

ビット	定義	設定
0	WDモード	(0)
1	パリティチップ	(1)
2	パリティ正モード	(0)
3	8/16ビットCPUとポートAインタフェース	(1)
4	反転ポートAアドレス0	(1)
5	反転ポートAアドレス1	(1)
6	検査合算折返し上げた上げ	(0)
7	リセット	(0)

チャンネル ファンクション

- 0:8 共通データバス550を経て各FIFO544に対し前後するこれらのチャンネル制御データの移動。FIFOが使用可能でありまた要求がそれから受信されると、チャンネルは実行可能になる。チャンネルが処理されると、完了のステータスが生成する。
- 11:10 ローカルデータバッファ564(後述)とVMEバス120との間のこれらのチャンネル制御データ移動。使用可能になると、チャンネルは実行可能になる。チャンネルが処理されると、完了のステータスが生成する。
- 12 使用可能になると、このチャンネルにより、ローカルデータバッファ564内のDRAMは、MPP524により生成したクロックにもとずいて復元する。この復元は、16列のバーストから成っている。このチャンネルは、完了のステータスを発生しない。
- 13 マイクロプロセッサの通信FIFO544は、このチャンネルにより処理される。使用可能が設定されたFIFO544が要求を主張すると、チャンネルは実行可能になる。このチャンネルは、完了のステータスを生成する。
- 14 マイクロプロセッサ510からVMEバス120への短い待ち時間の書き込みは、このチャンネルにより制御される。このチャンネルが使用可能になると、データは、特殊な32ビットレジスタ(後述)からVMEバス120へ移動する。このチャンネルは、完了のステータスを生成する。
- 15 これは、実行可能ステータスと完了ステータスが生成されない空白チャンネルである。

チャンネルは、最初に、よりクリティカルな要求の実行を可能にするため、順位づけられる。チャンネルの順位は、チャンネル14から始まる降順で割り当てられる。すなわち、すべてのチャンネルがサービスを要求している場合、チャンネル14は、処理される1番目のチャンネルである。共通データバス550は、双方向性レジスタ570を経て、36ビットシマン

特表平5-502525 (14)

クシオンバス572へ接続している。もう一つの双方向性レジスタ574は、ジャンクシオンバス572をローカルデータバス582に接続している。1メガバイトのDRAMを有し、パリティ付のローカルデータバッファ584は、ジャンクシオンバス572へ双方向に接続している。バッファ584は、バイトパリティ付で256K32ビットワードを有するように構成されている。SP114aは、ページモードでDRAMを起動して非常に早いデータ速度を支援する。これは、ランダム単ワードアクセスの代りに、データを連続転送することを必要とする。ローカルデータバッファ584は、安いディスクの重複配列(RAID)アルゴリズムを実行するために使用され、VMBバス120とSCSIバス540の一つにある周辺装置との間を直接の読取り及び書き込みするために使用されないことは明らかである。

すべてゼロを有する読取り専用レジスタ576もまた、ジャンクシオンバス572へ接続している。このレジスタは、システムメモリ116の大型ブロックのデータの転移、初期設定、クリアに主に使用される。

P1F0544と554、ローカルデータバッファ584、VMEバス120上のシステムメモリ116などの遠隔構成要素との相互間のデータ移動は、すべて、VME/P1F0 DMA制御装置580により制御される。VME/P1F0 DMA制御装置580は、NP110aのVME/P1F0 DMA制御装置272と似ており、付録書に述べられている。簡単に言えば、この装置は、ビットスライス型エンジン582とデュアルポートRAM584を備えている。デュアルポートRAM584の一つのポートは、32ビットマイクロプロセッサデータバス512を経てマイクロプロセッサ510と通信し、また、ほかのポートは、別の18ビットバスを経てビットスライス型エンジン582と通信する。マイクロプロセッサ510は、コマンド駆動をデュアルポートRAM584に設定し、チャンネルインポート580を使用して、VME/P1F0 DMA制御装置580へ信号を送り、コマンドにより実行させる。VME DMA制御装置は、チャンネルステータスとサービス要求を発生し、また終了ステータスをデュアルポートRAM584に戻す責務を有する。デュアルポートRAM584は、32ビットポートにおいて1K×32ビットとして、18ビットポートにお

いて2K×18ビットとして構成されている。マイクロプロセッサ510がVME/P1F0 DMA制御装置580を制御する方法を示す一例は次の通りである。最初に、所望のコマンドと所望のチャンネルの促進速度とを、デュアルポートRAMへ書き込む。例えば、コマンドは、「データブロックをP1F0544から、指定されたVMEアドレスで始まるシステムメモリ116のブロックへ転送する」ことである。第2に、マイクロプロセッサは、チャンネル実行可能ビットを所望チャンネルのチャンネル実行可能レジスタ580にセットする。

チャンネル実行可能ビットがセットされる時点では、対応するP1F0は、まだ、データを転送する準備が出来ていない。VME/P1F0 DMA制御装置580が、「準備完了」ステータスをチャンネルから受信したときのみ、制御装置580はコマンドを実行することとなる。この間に、制御装置580は、自由にコマンドを実行し、データをほかのチャンネルへ、または、そこから移動する。

DMA制御装置580が、「準備完了」ステータスを指定のチャンネルから受信すると、制御装置は、チャンネルコマンドと速度をデュアルポートRAM584から受信して、実行する。コマンドが完了すると、例えば、要求されたデータがすべて転送されると、DMA制御装置は、ステータスをデュアルポートRAM584へ書き戻し、チャンネルステータスレジスタ582のチャンネルに「完了」を表明する。次に、マイクロプロセッサ510は割り込みを、その時点で、チャンネルステータスレジスタ582を読み取り、どのチャンネルが割り込みかを決定する。次に、マイクロプロセッサ510は、対応するチャンネルのチャンネルインポートをクリアして、デュアルポートRAM584の終了チャンネルステータスをチェックする。

このようにして、高速度データ転送は、マイクロプロセッサ510により行われているほかの活動と完全に並行して、DMA制御装置580の制御のもとで行われる。データ転送は、マイクロプロセッサデータバス112と異なるバスを通して行われ、これによって、マイクロプロセッサ命令の取出しとの干渉を防止する。SP114aはまた、高速レジスタ580を有しており、これは、マイクロプロセッサデータバス112とローカルデータバス582との間に接続

している。高速レジスタ580は、最小のオーバーヘッドで、単一32ビットワードをVMEバス目標へ書き込むために使用される。このレジスタは、マイクロプロセッサから見て、書き込み専用である。ワードをVMEバス120へ書き込むために、マイクロプロセッサ510は、最初に、ワードをレジスタ580へ、また、所望のVME目標アドレスをデュアルポートRAM584へ書き込む。マイクロプロセッサ510がチャンネル実行可能レジスタ580の対応するチャンネルを実行可能にすると、DMA制御装置580は、データをレジスタ580から、デュアルポートRAM584に指定されたVMEバスアドレスへ転送する。次に、終了ステータスをデュアルポートRAMへ書き込み、チャンネル「終了」ビットをチャンネルステータスレジスタ582をセットする。

この手続は、単一ワードのデータの転送には非常に効率が良いが、大きいブロックデータには不効率になる。1ワードより大きいデータの転送は、一般にメッセージ送りの場合、P1F0554により通常行われる。

SP114aは、また、一連のレジスタ592を有しており、これは、NC110aのレジスタ282(図3)とPC112aのレジスタ382(図4)と似ている。これらのレジスタの詳細は、本発明を理解する上で重要ではない。

**記憶プロセッサ動作**

SP114の各々により支援される30個のSCSIディスクドライブは、構成によって、三つの大きい論理ディスクもしくは30個の独立SCSIドライブとして、ファイルコントローラ112の一つのようなクライアントプロセッサにより管理される。これらドライブが三つの管理ディスクとして管理される場合は、SPは、ディスクアーム結合を最小にするため九つの物理ドライブの各論理ドライブへデータを分配するようにRAID5の設計アルゴリズムを使用している。10個目のドライブは予備として残されている。RAID5アルゴリズム(redundant array of inexpensive drives, revision 5)は Patterson他により ACM SIGMOD Conference, Chicago, Ill., June 1-3 1987の "A Case For a Redundant Arrays of Inexpensive Disks (RAID)" にて説明されている。これは参考資料として添付されている。RAID5の設計では、ディスクデータはストライプに分割される。データストライプは八つの異なるディスクドライブに

順次、書き込まれる。九番目のパリティストライプ、即ち、八つのデータストライプの排他論理和、は九番目のドライブに書き込まれる。もし、ストライプのサイズが8Kバイトに設定されると、データ8Kの読み込みには一つのドライブのみを必要とする。データ8Kの書き込みには、データドライブおよびパリティドライブの二つのドライブを必要とする。書き込みは各パリティストライプを生成するために旧データを読み込む必要があるため、書き込みは、修正書き込みとも呼ばれる。SPは114aは、九つのSCSIドライブに對する九つの小規模読み込みの支援を同時に行う。ストライプのサイズが8Kに設定されると、データ84Kの読み込みには八つのSCSIドライブ全部が参加し、各ドライブがデータの8Kストライプを読み込む。並列動作は呼出クライアントには見えな

い。パリティストライプは、書き込み動作時、ドライブ結合を避けるため、九つのドライブ間で回転する。パリティストライプは、データの可用性を高めるために用いられる。ドライブの一つが故障すると、SP114aはパリティストライプより、ミッシングデータを再現する。このような場合は、SPセクター回復モードで動作する。不良ドライブの修復後、SPは、システムがオンラインにある間、修復完了ドライブに於いてデータを回復するような命令を受け付ける。

SP114aが30個の独立SCSIドライブを付加すると、パリティストライプは生成されず、クライアントは各ドライブを直接、アドレス指定する。

SP114aは、複数メッセージ(トランザクション、コマンド)を、200メッセージ/秒で、一掃に処理する。SPは114aは、初期システム構成後は、いかなるメッセージも起動しない。以下のSP114aの動作が定義されている。

- 01 ノーオペレーション
- 02 構成データ送信
- 03 構成データ受信
- 04 セクター読み込みおよび書き込み
- 05 キャッシュページ読み込みおよび書き込み
- 06 IOCTLオペレーション
- 07 SP114aローカルデータバッファのダンプ

特表平5-502525 (15)

- 00 SCSIドライブの起動/停止
- 0C 限会
- 0E メッセージログバッファの読みとり
- 0F SPI14aの割り込みセット

上記トランザクションは MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE の既述の資料で上記の名前の示すアプリケーションで詳細に説明される。本発明の理解には、上記の内二つのコマンド、即ち、セクター読み込みおよび書き込みならびにキャッシュページ読み込みおよび書き込みの構造と動作の説明が有益であろう。

セクター読み込みおよび書き込み

このコマンドは通常、FC112により発せられ、SPI14aがシステムメモリの指定ブロックとSCSIディスクの間合ったセクターの指定のシリーズとの間のデータ転送を実行するように作動する。ファイルコントローラ112に関して既に述べたように、個々のセクターは後述項目で識別される。特に、個々のディスクセクターはSCSIチャンネル番号(0-9)、指定ドライブのセクターアドレスで始まる。そのチャンネル番号のSCSI ID(0-2)、および読み込みまたは書き込みのセクター番号のカウントにより識別される。SCSIチャンネル番号は、SPI14aがRAID5で動作している場合は、ゼロである。

SPI14aは、30個のSCSIDライブで同時に30までのメッセージを実行する。コマンドFIFO534に現れるとすぐにマイクロプロセッサ510により処理される。SPI14aへのコマンドの大半とは異なって、読み込みおよび書き込みセクターコマンド(または読み込みおよび書き込みキャッシュメモリコマンドもまた) 初めに分類され、待ち行列にいれられる。従って、これらは通常順に処理されない。

ディスクアクセスコマンドが到着すると、マイクロプロセッサ510はどのディスクドライブを目標とするかを決定し、そのメッセージを、目標セクターアドレスにより分類されたそのディスクドライブのための待ち行列に挿入する。マイクロプロセッサ510は各ディスクドライブの待ち行列に存在する順序によって、

待ち行列のコマンドを実行する。ディスクアームの移動を最小限にするため、マイクロプロセッサ510はエレベーターの様な動作で待ち行列エントリの間を行き来する。

SCSIディスクドライブより何のエラー状態も検出されない場合は、コマンドは通常通り実行完了する。データチェックエラー状態が発生し、SPI14aがRAID5に構成されている場合は、冗長データを用いて、回復動作が自動的に開始される。SPI14aがRAID5に構成されて、ドライブが故障した場合、データチェック回復に類似した回復動作が行われる。

読み込み/書き込みキャッシュページ

このコマンドは、多重VMEアドレスがディスクデータをシステムメモリー118からまたはシステムメモリー118へ転送するために提供されていること以外には、読み込みおよび書き込みセクターと同様である。各VMEアドレスはシステムメモリー118でのキャッシュページを示し、このサイズもまたこのコマンドにて指定される。ディスクよりシステムメモリー118へデータを転送するとき、データは異なるキャッシュページへ分散され、ディスクへデータを書き込むときは、システムメモリー118内の異なるキャッシュページから集められる。従って、この動作は分散集積動作と呼ばれる。

SCSIディスク上の目標セクターは、読み込みおよび書き込みセクターコマンドのためにそれらが指定されと同様に、コマンドに於いて後述項目で識別される。コマンドの終了時のエラー状態の有無に関しては、読み込みおよび書き込みセクターコマンドと同様である。

DMAコントローラ580のデュアルポートRAMは、ビットスライスエンジン582により制御される各チャンネル用の分離コマンドの組を記憶する。各チャンネルはその前の状態を終了すると、マイクロプロセッサ510は、ディスクエレベーター待ち行列上の次のオペレーションを満足するために、そのチャンネルのデュアルポートRAM584へ新規DMAオペレーションを書き込む。

DMAコントローラ580に書き込まれたコマンドは、オペレーションコードならびに、非ブロックモード、標準VMEブロックモードまたはエンハンスドブロックモードにてそのオペレーションが実行されるかどうかを示すコードを

含む。DMAコントローラ580がサポートしているオペレーションコードを以下に示す。

オペレーションコード

- 0 ノーオペレーション
- 1 ゼロ→バッファ      ゼロレジスタ576よりゼロをローカルデータバッファ584に転送
- 2 ゼロ→FIFO      ゼロレジスタ576よりゼロを共通データバス上で現在選択されているFIFOへ転送
- 3 ゼロ→VMEバス      ゼロレジスタ576よりゼロをVMEバス120へ転送。システムメモリー118のキャッシュバッファを初期化するの用途。
- 4 VMEバス→バッファ      VMEバス120からデータをローカルデータバッファ584へ転送。このオペレーションは、冗長発生に参入するために、ダウンドライブ用の目標データをバッファに転送するために、書き込み期間中に使用される。RAID5のアプリケーションに対してのみ用いられる。
- 6 VMEバス→FIFO      VMEバスより新規データがドライブに書き込まれる。RAID5では、ローカルデータバッファ584で破損記憶されたデータから冗長データを発生することが必要なので、このオペレーションは、SPI14aがRAID5に構成されていない場合にのみ、使用される。
- 8 VMEバス→バッファおよびFIFO      目標データはVMEバス120よりSCSIデバイスへ転送され、また冗長発生に参入す

るためにローカルデータバッファ584で破損される。SPI14aがRAID5のオペレーションに構成されている場合のみ、使用される。

- 7 バッファ→VMEバス      このオペレーションは使用されない。
- 8 バッファ→FIFO      参入データは、ディスクドライブにて冗長データ、即ち、回復データを生成するために転送される。RAID5のアプリケーションでのみ使用される。
- 9 FIFO→VMEバス      このオペレーションは目標データを復検、ディスクドライブからVMEバス120へ転送するのに使用される。
- A FIFO→バッファ      参入データを、回復、修正オペレーションに使用するため、転送。RAID5のアプリケーションでのみ使用される。
- B FIFO→VMEバスおよびバッファ      このオペレーションはデータ回復に参入するため、目標データをセーブするのに使用される。RAID5のアプリケーションでのみ使用される。

システムメモリー

図6はシステムメモリー基板118aの一つの好適なアーキテクチャーの図解ブロック図である。他のシステムメモリー基板の各々もこれと同じである。各メモリー基板は、エンハンスドVMEバス120上でスレーブとして動作するので、基板上でCPUを必要としない。タイミングコントロールブロック810は、必要とするスレーブコントロールのオペレーションを提供するに十分でなければならぬ。とくに、エンハンスドVMEバス120のコントロール部からのコントロール信号に反応して、タイミングコントロールブロック810は、エンハンスドVMEバス120とマルチプレクサユニット814の間で32ビット



特表平5-502525 (18)

トのデータの適切な転送方向設定のために、32ビットのパワー812をイネーブルする。データの転送方向により、マルチプレクサー814は、8Mビット×72ビット語メモリアレー820のためのマルチプレクスおよびマルチプレクス復元を有する。エラー訂正コード発生(ECC)およびテストユニット822は、マルチプレクサー814へ接続され、またこの場合も、転送方向によって、8ビットのECCデータを発生するかまたは確認する。ECC確認の状況はタイミングコントロールブロック810へ送り返される。

エンハンスドVMEバスプロトコル

VMEバス120は物理的には、通常のVMEバスと変わらないが、NCおよびSPの各々は、エンハンスドVMEブロック転送プロトコルを用いてデータを送信するための付加回路およびファームウェアを含む。エンハンスドプロトコルは ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFERの要約の示すアプリケーションに詳細に、説明されていて、この付録にその要約が載せられている。代表的には、LNFSファイルデータを、NCとシステムメモリー間、SPとシステムメモリー間で転送、ならびにシステムメモリーを経由して一つのNCからもう一つのNCへ経路指定されているパケットの転送がサーバー100でエンハンスドプロトコルを使用する唯一の転送のタイプである。VMEバス120上の他のすべてのデータ転送は従来のVMEブロック転送プロトコルまたは通常ブロック転送プロトコルを用いている。

メッセージバス

上記で明らかのように、サーバー100の異なるプロセッサは互いにあるタイプのメッセージで通信する。ソフトウェアでは、これらのメッセージは、メッセージカーネルにより取り扱われ、既に述べた MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE アプリケーションに詳述されている。ハードウェアでは、以下のように実現されている。

NC110の各々、FC112の各々、およびSP114の各々はNC110aの290のようなコマンドPIFO、即ち、通信PIFOを含む。ホスト118もまたコマンドPIFOを含むが、このホストが未改造の購入品プロセッサ蓋

板なので、PIFOはソフトウェア上でアミューレートされる。各プロセッサのコマンドPIFOの書き込みポートはVMEバス120上で他のどのプロセッサからも同様、アドレス指定できる。

同様に、SP114を除き、プロセッサの各々はNC110aのCPUメモリー214のような共用メモリーを含む。この共用メモリーもまたサーバー100の他のプロセッサのどれからもアドレス指定可能である。

もし、一つのプロセッサ、例えば、ネットワークコントローラー110aが、第二のプロセッサ、例えば、ファイルコントローラー112aにメッセージもしくはコマンドを送信すると、次のようになる。第一に、自身の共用メモリー(例えば、NC110aのCPUメモリー214にて)にメッセージを形成する。第二に、送信プロセッサ側のマイクロプロセッサが直接、メッセージ記述子を、受信プロセッサ側のコマンドPIFOに書き込む。コマンドがネットワークコントローラー110aよりファイルコントローラー112aへ送られるように、マイクロプロセッサ210は、書き込みを、NC110aのパワー284と、ファイルコントローラー112aのVMEバス120およびパワー384により実行する。

コマンド記述子は、その上位30ビットに、送信側の共用メモリーでの4倍長合わせされたメッセージの始まりを示すVMEアドレスを含む32ビット語である。下位2ビットは、次の如く、メッセージのタイプを示す。

タイプ	内容
0	送信されている新規メッセージを示すポインター
1	応答メッセージを示すポインター
2	正方向へ転送するメッセージを示すポインター
3	解放されるメッセージを示すポインター；またはメッセージ肯定応答

全メッセージは128バイト長である。

受信プロセッサがそのコマンドPIFOのコマンド記述子に達すると、直接、送信側の共用メモリーにアクセスし、それを受信側自身のローカルメモリーにコピーする。コマンドをネットワークコントローラー110aからファイルコ

ントローラー112aに送出するには、パワー284、VMEバス120およびパワー384を経由しFC CPUメモリー314へ向かう。NC CPUメモリー214からの通常VMEブロックモード転送もしくは非ブロックモード転送による方法がある。FCマイクロプロセッサ310は、VMEバス120上でこの目的のために、直接、NC CPUメモリー214へアクセスする。

受信プロセッサがコマンドを受信し、その仕事を実行すると、応答メッセージを送信プロセッサに送る。応答メッセージは、無修正のものコマンドメッセージでも良いし、もとのメッセージを修正したメッセージ、あるいは、まったく新しいメッセージでも良い。もし、応答メッセージがもとのコマンドメッセージと同一でないなら、受信プロセッサは、もとの送信側の共用メモリーに直接、アクセスし、もとのコマンドメッセージを修正するかあるいは完全にもとのコマンドメッセージを上書きする。FC112aよりNC110aへの応答は、パワー384、VMEバス120、パワー284を経由し、そしてNC CPUメモリー214への、通常VMEブロックモード転送もしくは非ブロックモード転送を必要とする。ここでも、FCマイクロプロセッサ310は、VMEバス120上でこの目的のために、直接、NC CPUメモリー214へアクセスする。

もとのコマンドメッセージが変更されたかどうかによって、受信プロセッサは、応答メッセージ記述子を直接、もとの送信側コマンドPIFOに書き込みをする。応答メッセージ記述子は、もとのコマンドメッセージ記述子と同一のVMEアドレスを含み、その一部の下位2ビットは、これが応答メッセージであることを示すように変更される。FC112aよりNC110aへの応答の目的で、メッセージ記述子の書き込みは、パワー384、VMEバス120およびNC上のパワー280を経由して、コマンドPIFO290へ直接アクセスするマイクロプロセッサ310によりなされる。これが、いったんなされると、受信プロセッサは、コマンドメッセージのコピーを記憶しているそのローカルメモリーのパワーを、解放する。

もとの受信プロセッサがそのコマンドPIFOの応答メッセージ記述子に達すると、そのメッセージの送信を可能としたもとのプロセスを呼び直し、それ

を解放させる。応答メッセージを解放した後、もとの送信プロセッサは、それ自身のローカル共用メモリーのもとのコマンドメッセージパufferを解放する。

以上述べたように、ネットワークコントローラー110aはVMEバス120メッセージ記述子を書き込むためにパワー284データ経路を使用し、VMEバス120よりのメッセージをCPUメモリー214にコピーするため、パーティPIFO270と共にVME/PIFO DMAコントローラー272を使用する。他のプロセッサは、パワー284データ経路を使用して、CPUメモリー214から読み込みをする。

ファイルコントローラー112aは、パワー384のデータ経路を利用して、メッセージ記述子をVMEバス120へ書き込み、同じデータ経路を使用して、他のプロセッサの共用メモリーからメッセージをコピーする。両者ともにマイクロプロセッサ310のコントロールのもとに、実行される。ほかのプロセッサまたはパワー384データ経路により、CPUメモリー314からメッセージをコピーする。

記憶メモリー114aは、上記のような方法で、高速レジスタ590を用いて、メッセージ記述子をVMEバスへ書き込み、DMAコントローラー580およびPIFO554を利用して、他のプロセッサからメッセージをコピーする。SP114aは共用メモリーを有さないで、その機能を実行するため、システムメモリー118のパufferを使用する。即ち、メッセージ記述子を他の一つのプロセッサコマンドPIFOへ書き込む前に、SP114aは、第一にDMAコントローラー580およびPIFO554を用いて、システムメモリー118の以前自身に割り付けられたパufferにメッセージをコピーする。メッセージ記述子に含まれるVMEアドレスは、システムメモリー118に設けるメッセージのVMEアドレスを反映する。

本発明は特定の実施例を用いて説明して、従って、多くの変形、変更が、本発明の範囲を越えないで可能であること表明であらう。

特表平5-502525 (17)

付録A

VME/PFIFO DMAコントローラ

記憶プロセッサ114aで、DMAコントローラ580はマイクロプロセッサ510の制御のもとデータ通路を管理している。DMAコントローラ580は82.5ns毎にバイブライン命令を実行するマイクロコード化された18ビットのビットスライスデバイスである。これはチャンネルステータス582を発生し、マイクロプロセッサ510によってデュアルポートのRAM584に記憶されるパラメータを持つリクエストを処理する責任がある。終了ステータスはRAM584に送られ、割り込みがマイクロプロセッサ510にたいして発生する。

**制御記憶** 制御記憶は、DMAコントローラ580を制御するマイクロコード化された命令を含む。制御記憶は、1K×48ビットマイクロワードとなるように構成された8個の1K×8 PROMよりなる。制御記憶内の位置はシーケンサによりアドレス指定され、データはバイブラインレジスタの人力部に提供される。

シーケンサ

シーケンサはバイブラインデータおよび種々のステータスビットに基づいた制御記憶アドレスを発生することによりプログラムフローを制御する。制御記憶アドレスは10ビットよりなる。制御記憶アドレスの8:0ビットは、ALU出力もしくはインクリメンタの出力を、その入力とするマイクロプロセッサから得られる。インクリメンタは、バイブラインレジスタの8:0ビットでプレローディングすることができるか、またはテストコンディションの結果として、増分することができる。1Kアドレス範囲は、ラッチフラグで二つのページに分断され、マイクロプログラムがいずれのページからも実行されるようになっている。プランチは、しかし、選択されたページ内にある。条件順序付けは、バイブラインが与えたアドレスをテストコンディションが辨分するようにして実行される。偽コンディションによりバイブラインアドレスよりの実行がなされ、真コンディションによりアドレス+1よりの実行がされる。ALUの出力はルーチンへ直接、向けるかまたは呼出ルーチンへ復帰するかするためにアドレスソースとし

て、選択される。サブルーチン呼び出すときは、呼出ルーチンは同じページになければならない。復帰時、サブルーチンまたは関連したページが選択されるからである。

**ALU** ALUは単一の乗算回路1DT49C402Aを含む。これは18ビットで84のレジスタを持つ四つの2901にも類似している。ALUは主に、インクリメンティング、デクリメンティング、加算、ビット操作に作用される。必要なすべての制御信号は制御記憶で生成される。ここに、参考資料として引用した、1DT HIGH PERFORMANCE CMOS 1988 DATA BOOKはALUに関して、さらに付加的な情報を載せている。

**マイクロワード** 48ビットのマイクロワードは、DMAコントローラ580の種々の機能を制御するいくつかのフィールドを含む。マイクロワードのフォーマットが定義され、ニーモニックおよび各機能の説明と共に以下に示される。

AI<8:0>	47:39	(ALU命令ビット8:0) AIビットは49C402A ALU用に命令を提供する。ALUの命令の完全な定義については1DTデータブックを参照。40C402Aの19の信号入力は常時ロウであることに注意。
CIN	38	(キャリー入力) このビットはキャリー入力をALUに強制している。
RA<5:0>	38:32	(レジスタ-Aアドレスビット5:0) これらのビットは84レジスタの一つをALUの“A”オペランドとして選択する。これらのビットはまた、ALUバスのリテラルビット15:10を提供する。
RB<5:0>	31:26	(レジスタ-Bアドレスビット5:0) これらのビットは84レジスタの一つをALUの“B”オペランドとして選択する。これらのビットはまた、ALUバスのリテラルビット9:4を提供する。

LTD	25	(ラッチフラグデータ) セットされると、このビットは選択されたラッチフラグをセットするように働く。リセットされると、このビットは選択されたラッチフラグがクリアされるように働く。このビットもまたALUバスのためのリテラルビット8として機能する。
LFS<2:0>	24:22	(ラッチフラグ選択ビット2:0) これらビットの意味は、ALUバスに何がソースとして選択されるかに依存する。バスソースとしてリテラルフィールドが選択されると、LFS<2:0>はリテラルビット<2:0>として動作し、それ以外はこれらビットはラッチフラグの一つを選択するの用にられる。

LFS<2:0>

選択フラグ

0	この値は空フラグを選択する。
1	セットされると、このビットはバファークロックをイネーブルにする。リセットされると、バファークロックをディスエーブルにする。
2	このビットがクリアされると、VMEバス転送、バファーマネージメントおよびRASがすべてディスエーブルされる。
3	使用されていない。
4	セットされると、このビットはVMEバス転送をイネーブルにする。
5	セットされると、このビットはバファーマネージメントをイネーブルにする。
6	セットされると、このビットは行アドレストロープをDRAM/バファーマネージメントにアサートする。
7	セットされると、このビットは制御記憶のペ

SRC<1:0>	20:21	ページ0を選択する。(ALUバスソース選択ビット1:0) これらのビットはデータソースを選択し、ALUバスでイネーブルされるようにする。
----------	-------	--

SRC<1:0>

選択ソース

0	ALU	
1	デュアルポートRAM	
2	リテラル	
3	予約:未定義	
PR<2:0>	19:17	(バスフラグ選択ビット2:0) これらビットはバス出力されるフラグ/信号を選択する。

PF<2:0>

フラグ

0	空	
1	SGL_GLK バファークロックの単一クロックの転送	
2	SET_VB VMEおよびバファーマネージメントをセットに強制する。	
3	CL_PERR バファーマネージメントエラーステータスをクリアする。	
4	SET_DN 現在選択中のチャンネルのチャンネルダグンステータスをセットする。	
5	INC_ADR デュアルポートRAMアドレスを増分する。	
6:7	予約:未定義	
BEST<3:0>	18:13	(宛先選択ビット3:0) これらビットは

特表平5-502525 (18)

10の宛先の一つをALUバスからロードするように選択する。

DEST<3:0>	宛先
0	空
1	WR_RAM デュアルポートRAMへALUバスのデータを書き込ませる。 D<15:0> -> RAM<15:0>
2	WR_BADD ALUバスからデータをDRAMアドレスカウンタへロードする。 D<14:7> -> mux addr<8:0>
3	WR_VADL ALUバスからデータをVMEアドレスレジスタの最下位2バイトへロードする。 D<15:2> -> VME addr<15:2> D1 -> ENB_レジスタ D<15:2> -> VME addr<15:2> D1 -> ENB_ENH D0 -> ENB_BLK
4	WR_VADH VMEアドレスレジスタの最上位2バイトをロードする。 D<15:0> -> VME addr<31:16>
5	WR_RADD デュアルポートRAMアドレスカウンタをロードする。 D<10:0> -> ram addr<10:0>
6	WR_WCNT

5	ALU_NEG	-ALUネガティブ
6	XFR_DONE	-低送完了
7	PAR_ERR	-パフォーマリティーエラー
8	TIMOUT	-バスオペレーションタイムアウト
9	ANY_ERR	-エニーエラーステータス
14:10	予約:未定義	
15	CH_RDY	-ネクストチャンネルレディ

NEXT\_A<8:0> 8:0 (ネクストアドレスビット8:0) 制御記憶の現在のページより命令を選択し、実行する。

**デュアルポートRAM** デュアルポートRAMは、それによってDMAコントローラ-580とマイクロプロセッサ-510の間で、コマンド、パラメータ、およびステータスが通信される媒体である。RAMはマスターポートでは1Kx32として構成され、DMAポートでは2Kx18として構成される。RAMの書き込みおよび読み込みはいずれのポートでも可能である。

RAMは、11ビットのアドレスをアドレスカウンタにロードして、DMAコントローラ-580によりアドレス指定される。データは双方向レジスタに読み込まれ、アドレスカウンタは次の位置を渡すために、再分される。

RAMの書き込みは、RAMアドレスをロードした後、データをプロセッサからレジスタにロードすることによって、なされる。このような書き込みは他のすべてのプロセッササイクルで実行される。

RAMはカウントブロックポインタ、終了ステータス、高速バスアドレスおよびパラメータブロックを含む。以下は、RAMのフォーマットである。

ワードカウンタをロードする。

D15 ->カウントインヘブル\*

D<14:8>->カウント<6:0>

7 WR\_CO  
コチャネル選択レジスタをロードする。

D<7:4>->CO<3:0>

8 WR\_NXT  
ネクストチャンネル選択レジスタをロードする。

D<8:9>->NEXT<3:0>

9 WR\_CUR  
カレントチャンネル選択レジスタをロードする。

D<3:0>->CURR<3:0>

10:14 予約:未定義

15 JUMP

制御記憶シーケンサにALUデータバスを選択させる。

D<8:0>->CS\_A<8:0>

TEST<3:0> 12:9 (テストコンディション選択ビット3:0) インクリメンタに付するキャリー入力として、テストマルチプレクサ-にたいする16の入力の一つを選択する。

TEST<3:0> コンディション

0	偽	-常に偽
1	真	-常に真
2	ALU_COUNT	-ALUのキャリー出力
3	ALU_EQ	-ALU出力と同一化
4	ALU_OVR	-ALUオーバーフロー

オフセット	31	0
0	カレントポインタ-0	ステータス0
4	イニシャルポインタ-0	
58	カレントポインタ-B	ステータスB
5C	イニシャルポインタ-B	
80	不使用	不使用
84	不使用	不使用
88	カレントポインタ-D	ステータスD
8C	イニシャルポインタ-D	
70	不使用	ステータスE
74	高速バスアドレス31:200	
78	パラメータブロック0	
??	パラメータブロックn	

イニシャルポインタはチェインの第1コマンドブロックを示す32ビットの値である。カレントポインタは、カレントコマンドブロックを示すためにDMAコントローラ-580によって使用される16ビットの値である。カレントコマンドブロックポインタは、チャンネルモイネブルする前に、マイクロプロセッサ-510によって0x0000に初期化されなければならない。カレントブロックポインタにて0x0000を検出すると、DMAコントローラ-580は、イニシャルポインタよりカレントポインタに下位16ビットをコピーする。DMAコントローラ-580がパラメータブロックの指定オペレーションを完了すると、カレントポインタは次のブロックを示すように更新される。パラメータブロックがもうない場合には、ポインタは0x0000にセットされる。

ステータスバイトは、実行された最終チャンネルオペレーションの終了ステータスを示す。以下のステータスバイトが定義されている。

特表平5-502525 (19)

ステータス	意味
0	エラー無し
1	接続オペレーションコード
2	バスオペレーションタイムアウト
3	バスオペレーションエラー
4	データ経路パリティエラー

パラメータブロックのフォーマット

オフセット 31 0

0	順方向リンク	
4	不使用	ワードカウント
8	VMEアドレス31:2, ENH, BLK	
C	TERM 0 OP 0	BUF ADDR 0
	.	.
	.	.
	.	.
C+(4Xn)	TERM n OP n	BUF ADDR n

順方向リンク-順方向リンクは次のパラメータブロックの第一語を、実行するために示す。また、これにより、いくつかのパラメータブロックを初期化され、実行用のオペレーションのシーケンスを生成するように連環される。順方向ポインタは次のフォーマットを有す。

A31:A2, 0, 0

バファアアドレスを指定する。16ビットのみが、1Mバイトバッファのために、利用できる。従って、開始アドレスは常時、18バイト境界の範囲に入る。プログラマーは開始アドレスはモデュロ128バイト境界に従うことを確実にしなければならない。バファアアドレスは、各データバーストの完了時、DMAコントローラ-580によって更新される。

|A18|A18|A17|A18|A15|A14|A13|A12|A11|A10|A9|A8|A7|A8|A5|A4|

TERM-パラメータブロックの最終バファアアドレスおよびオペレーションはターミナルビットにより、識別される。DMAコントローラ-580は、このビットに出会うまで、バファアアドレスおよびオペレーションをフェッチし続ける。いったん、パラメータブロックの最終オペレーションが完了すると、ワードカウンタは更新され、もしゼロに等しくないなら、オペレーションのシリアズが繰り返される。ワードカウンタがゼロとなると、以下の順方向リンクポインタが次のパラメータにアクセスするために、使用される。

0 0 0 0 0 0 0 T

OP-オペレーションはopコードで指定される。opコードバイトは次のフォーマットを有する。

0 0 0 0 OP5 OP2 OP1 OP0

opコードは以下に列挙される ("FIFO" はFIFO644または554のどの一つでもかまわない)。

このフォーマットは、パラメータブロックが4倍長バイト境界で開始することを規定している。ポインタ-0x00000000は順方向リンクが存在しないことを示す特別例である。

ワードカウンタ-ワードカウンタは、各バファアアドレスよりまたは各バファアアドレスへ、ならびに、VMEアドレスよりまたはVMEアドレスへ転送される4倍長バイトワードの数を示す。86Kワードのワードカウンタは、ワードカウンタをゼロで初期化して、指定される。ワードカウンタは次のフォーマットを持つ。

ID15|D14|D13|D12|D11|D10|D9|D8|D7|D6|D5|D4|D3|D2|D1|D0|

ワードカウンタは、最後に指定されたバファアアドレスよりまたは最後に指定されたバファアアドレスへの転送の完了時、DMA580により更新される。ワードカウンタは、各バファアアドレスよりまたは各バファアアドレスへの転送完了時には、更新されず、従って、バファアへのまたはバファアよりの転送の総データ量の正確な指標にはならない。ワードカウンタはVMEバスへまたはFIFO544または554の一つへ転送されたデータの量を意味する。

ENH-このビットがセットされると、上述の ENHANCED VMEBS PROTOCOL UTILIZING PSEUDOSYNC CHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER アプリケーションに説明されている、VMEバス転送時に使用されるエンハンスドブロック転送プロトコルを選択する。24ビットまたは16ビットのアドレススペースからまたは24ビットまたは16ビットのアドレススペースよりの転送を実行しているとき、開始アドレスが8バイトに整列されていないとき、あるいは、ワードカウンタが偶数でないとき、エンハンスドプロトコルは自動的にディスエーブルされる。

BLK-セットされていると、このビットは、VMEバス転送時に使用される従来のVMEブロックモードプロトコルを選択する。ブロックモードは、16ビットスペースよりまたは16ビットスペースへ転送が実行されているときは、自動的にディスエーブルされる。

BUF ADDR-バファアアドレスは、隣接のオペレーションのため、開始

OPコード	オペレーション
0	ノオペレーション
1	ゼロ -> バファア
2	ゼロ -> FIFO
3	ゼロ -> VMEバス
4	VMEバス->バファア
5	VMEバス->FIFO
6	VMEバス->バファアおよびFIFO
7	バファア ->VMEバス
8	バファア ->FIFO
9	FIFO ->VMEバス
A	FIFO ->バファア
B	FIFO ->VMEバスおよびバファア
C	保留
D	保留
E	保留
F	保留

特表平5-502525 (20)

付図B

エンハンスドVMEブロック転送プロトコル

エンハンスドVMEブロック転送プロトコルは、マスター機能モジュールと、データ転送バスにより理論的に接続されるスレーブモジュールとを有するVMEバックプレーンバスでの使用を想定したVMEバスコンパチブル類似同期高速転送ハンドシェイクプロトコルである。データ転送バスは、データストロープ信号ラインおよびデータ転送肯定応答信号ラインを含む、ハンドシェイクを実行するために、マスターはデータストロープライン上で、一定の間隔のデータストロープ信号を送信する。マスターは次に、データ転送肯定応答信号ライン上で、スレーブモジュールより、データ転送肯定応答信号の受信を待つ。スレーブは、データ転送肯定応答信号ライン上で、一定の間隔のデータ転送肯定応答信号を送信することによって、応答する。

ハンドシェイクプロトコルの擬似同期性に矛盾しないように、転送データは、転送オペレーションが書き込みか読み込みかによって、ただ一つの信号に対して参照される。

データをマスター機能ユニットからスレーブに転送する場合、マスターは送信するデータを同時送信する。マスターはデータストロープ信号をアサートし、スレーブは、そのデータストロープ信号に反応して、マスターによって同時送信されたデータを捕獲する。同時に、スレーブからマスターにデータを転送する場合、スレーブはマスターユニットに転送すべきデータを同時送信する。スレーブはデータ転送肯定応答信号をアサートし、マスターは、このデータ転送肯定応答信号に反応して、スレーブによって同時送信されたデータを捕獲する。

高速転送プロトコルは、本発明にとって不可欠ではないが、データ転送速度を大幅に高速化して、VMEバスバックプレーンバスを横断する多量のデータの高速転送を容易にする。これらデータ速度は、データストロープ信号およびデータ転送肯定応答信号を概念的に切り取られたハンドシェイクを用いることによって、また、全データおよびコントロールラインの高位カレントドライバーを指定することによって達成される。

データ転送のエンハンスド擬似同期法（以後「高速転送モード」と呼称）は

似同期として特徴づけられる。高速転送モードプロトコルは、DS0\*が、スレーブよりの応答に無関係にアサートされ、デアサートされるということによれば、同期方式である。高速転送モードプロトコルの非同期の特徴は、マスターが、前のストロープに対する応答がスレーブより受信されるまで、次にDS0\*をアサートしないということによる。結果、このプロトコルは同期および非同期成分を持つので、正確には「擬似同期」として分類される。

高速転送プロトコルのブロック書き込みサイクル期間のデータの転送はDS0\*に対してのみ参照される。マスターはまず有効なデータをスレーブに同時送信し、DS0\*をスレーブに対してアサートする。スレーブには、その間データを捕獲するDS0\*のアサートの後、あらかじめ設定された期間が与えられる。従って、スレーブモジュールは、いかなるときも、データ捕獲に備えなければならない。DTACK\*は、転送サイクル中に、参照されないからである。

同時に、高速転送プロトコルのブロック読み込み期間のデータ転送はDTACK\*に対してのみ参照される。マスターはまずDS0\*アサートする。スレーブはマスターにデータを同時送信し、次にDTACK\*をアサートする。マスターには、その間データを捕獲するDTACK\*のアサートの後、あらかじめ設定された期間が与えられる。従って、DS0\*は転送サイクルで参照されないため、マスターモジュールはいかなるときもデータを捕獲する用意がなければならない。

図7のAからCまでは、高速転送プロトコルブロック書き込みサイクルを実行するオペレーションを示すフローチャートである。ブロック書き込みサイクルを開始するために、マスターは転送するデータのメモリアドレスおよびアドレス変更をDTB\*バスにより同時放送する。マスターはまた割り込み応答信号(IACK\*)をトイにし、LWORD\*信号をロウ701にする。マスターによって同時送信される特別なアドレス変更子、例えば「1F」、は、スレーブモジュールに対して、高速転送プロトコルはブロック書き込みを実行するために使用されるということを指示している。

転送データの開始メモリアドレスは、84ビット境界になければならない。転送データのブロックのサイズは、84ビットの倍数でなければならない。VMEバス標準に準拠するためには、ブロックは、新規アドレスサイクルを実行しない

IEEE VMEバックプレーンバス規格に合致するように実現される。このプロトコルは、高速転送モードの使用していることを示すために、VMEバス標準で定義された、ユーザー定義のアドレス変更子を使用する。標準VMEバスプロトコル実現可能であるだけの、従来のVMEバス機能ユニットは、高速転送モードを用いてなされる転送を捕獲するので、結果として、高速転送モードを実現する能力のある機能ユニットと完全にコンパチブルである。高速転送モードは、ハンドシェイクを実行するのに必要なバス伝播の数を、従来のVMEバスプロトコルで必要な四つより、単に二つに減じる。同様に、ブロック読み込みおよびブロック書き込みデータ転送を実行するのに必要なバス伝播の数も減らすことが可能である。ハンドシェイクおよびデータ転送機能を進取するVMEバスを通過する伝播を減らすことによって、転送速度は大幅に高められる。

エンハンスドプロトコルは上述の ENHANCED WEBUS PROTOCOL アプリケーションに詳述されており、ここで簡単に説明する。従来のVMEバス標準の知識はあるものと想定する。

高速転送モードハンドシェイクプロトコルでは、ハンドシェイクを実行するために二つのバス伝播のみが使用されて、従来のプロトコルによる四つとは異なる。データ転送サイクルの開始にて、マスターは、ある一定の間隔のバスの形で、DS0\*をアサートし、デアサートする。DS0\*のデアサートはスレーブより応答が受信されたかどうかにかかわらず実行される。マスターは次にスレーブより肯定応答を待つ。最後のDS0\*バスの後、応答DTACK\*信号がスレーブより受信されるまでは、生成されない。スレーブのDTACK\*のアサートを受信すると、マスターは、希望すれば、すぐにデータストロープを再アサートすることができる。高速転送モードプロトコルでは、最後のDS0\*のアサートの前提条件としては、マスターがスレーブによるDTACK\*のアサートを持つことを必要としない。高速転送モードでは、信号のリーディングエッジ（即ち、アサート）のみが有効である。このように、DS0\*またはDTACK\*のどちらかのデアサートはハンドシェイクの完了にまったく無関係である。高速転送プロトコルはDS1\*ラインをデータストロープの目的に、一切使用していない。

高速転送モードプロトコルは、同期および非同期の両方の面を有するので、接

で、256バイト境界を越えてはならない。

DTBに接続されているスレーブモジュールは、バス上でマスターによって同時送信されるアドレスおよびアドレス変更子を受信し、LWORD\*ロウおよびIACK\*ハイを受信する703。アドレスおよびアドレス変更子を同時送信したすぐ後701、マスターはAS\*信号をロウとする705。スレーブモジュールはAS\*ロウ信号を受信する707。各スレーブは、同時送信のアドレスはその各スレーブにとって有効かどうか決定することによってデータ転送に参加するか否かを個別に決定する709。もし、アドレスが有効でなければ、データ転送はそのスレーブに関係せず、それは残りのデータ転送サイクルを無視する。

マスターはWRITE\*をロウにして、実行されようとしている転送サイクルはWRITEオペレーションであることを指示する711。スレーブはRFB\*ロウ信号を受信し713。データ転送オペレーションがWRITEオペレーションであることを確認し、DS0\*信号ラインのハイからロウへの変化を示す信号の受信を待つ715。マスターはDTACK\*とBERR\*がハイになるまで待つ718。これは以前のスレーブがもはやDTBをドライブしていないことを意味する。

マスターは、転送データの第1セグメントをデータラインD00からD31までに配置できるようにする719。データをD00からD31に配置した後、マスターはDS0\*をロウとし721、あらかじめ設定された時間の後、DS0\*をハイとする723。

それぞれ721および723に対応する、DS0\*のハイよりロウへの変化に反応して、スレーブはデータラインD00からD31のデータライン上、マスターによって送信されているデータをラッチする725。マスターは、転送データの次のセグメントを、D00からD31へ配置し727、図7Bで示されるハイよりロウへの変化する信号の形としてのDTACK\*信号の受信を待つ729。

図7Bが示すように、スレーブはDTACK\*をロウとし731、あらかじめ設定された時間の後、DTACKをハイとする733。スレーブによってラッチされた、725、データは、データを記憶するように選択されたデバイスへ書き込まれる735。スレーブはまたデバイスアドレスを勘弁する735。スレーブは、

特表平5-502525 (21)

さらにまたほかのDS0\*のハイからロウへの変化を持つ737。

転送データのブロックの次のセグメントの転送を開始するため、マスターはDS0\*をロウ738とし、あらかじめ設定された時間の後、DS0\*をハイとする741。それぞれ738および741で示される、DS0\*のハイからロウへの変化に対応して、スレーブは、データラインD00からD31上でマスターによって同時通信されるデータをラッチする743。マスターは、データラインD00からD31上に、転送データの次のセグメントを配置し745、ハイよりロウへの変化の形をとるDTACK\*信号の受信を持つ747。

スレーブは次に、DTACK\*をロウとし、748、あらかじめ設定された時間の後、DTACK\*をハイとする751。スレーブによってラッチされた、743、データは、データを記憶するために選択されたデバイスに書き込まれ、そのデバイスは差分される753。スレーブはDS0\*のハイからロウへの次の変化を持つ737。

データの転送は、データの全てがマスターよりスレーブに転送されるまで、上記のように継続する。データの全てが転送されると、マスターはアドレスライン、アドレス変更ライン、データライン、IACK\*ライン、LWORD\*ラインおよびDS0\*ラインを解放する755。マスターはハイからロウへ変化するDTACK\*の受信を持つ757。スレーブはDTACK\*をロウにし759、あらかじめ設定された時間の後、DTACK\*をハイとする761。ハイからロウへ変化するDTACK\*の受信に対応して、マスターはAS\*をハイとし、768、次にAS\*ラインを解放する765。

図8のAからCは高速転送プロトコルブロック読み込みサイクルを実行するオペレーションを示すフローチャートである。ブロック読み込みサイクルを開始するため、マスターは、転送データのメモリアドレスおよびアドレス変更子をDTB\*バスで、同時通信する。マスターはLWORD\*信号をロウとし、IACK\*信号をハイとする801。すでに、述べたように、特別なアドレス変更子は、スレーブモジュールに、高速転送プロトコルがブロック読み込みを実行するのに使用されるということを表示する。

DTBに接続されているスレーブモジュールは、バス上でマスターによって同

時通信されるアドレスおよびアドレス変更子を受信し、LWORD\*ロウおよびIACK\*ハイを受信する803。アドレスおよびアドレス変更子を同時通信したすぐ後801、マスターはAS\*信号をロウとする805。スレーブモジュールはAS\*ロウ信号を受信する807。各スレーブは、同時通信のアドレスはその各スレーブによって有効かどうか決定することによってデータ転送に参加するか否かを個別に決定する809。もし、アドレスが有効ではなければ、データ転送はそのスレーブに關係せず、それは残りのデータ転送サイクルを無視する。

マスターはWRITE\*をハイにして、実行されようとしている転送サイクルはREAD\*オペレーションであることを表示する811。スレーブは、WRITE\*ハイ信号を受信し813、データ転送オペレーションがREAD\*オペレーションであることを確認し、データラインD00からD31へ転送データの第一セグメントを配置する819。マスターはDRACK\*とBERR\*がハイになるまで待ち818、これは以前のスレーブがもはやDTBをドライブしていないことを意味する。

マスターは次に、DS0\*をロウとし、821、あらかじめ設定された時間の後、DS0\*をハイとする823。マスターはDTACK\*信号ラインがハイよりロウとなるのを待つ824。図8Bに示されるように、スレーブはDTACK\*信号をロウとし、825、あらかじめ設定された時間の後、DTACK\*信号をハイとする827。

それぞれ825および827に対応する、DTACK\*のハイよりロウへの変化に対応して、マスターはデータラインD00からD31のデータライン上、スレーブによって通信されているデータをラッチする831。マスターによってラッチされている、831、データは、データを記憶するために選択されたデバイスに書き込まれ、そのデバイスは差分される833。

スレーブは転送データの次のセグメントをデータラインD00からD31へ配置し、828、次のDS0\*のハイからロウへの変化の待つ837。

転送データのブロックの次のセグメントの転送を開始するため、マスターはDS0\*をロウ838とし、あらかじめ設定された時間の後、DS0\*をハイとする841。マスターはDTACK\*ラインがハイよりロウへ変化するのを待つ

843。

スレーブはDTACK\*をロウとし、845、あらかじめ設定された時間の後、DTACK\*をハイとする847。

それぞれ838および841に対応する、DTACK\*のハイよりロウへの変化に対応して、マスターはデータラインD00からD31のデータライン上、スレーブによって通信されているデータをラッチする849。マスターによってラッチされている、845、データは、図8Cが示すようにデータを記憶するために選択されたデバイスに書き込まれ、851、そのデバイスアドレスは差分される。スレーブは、転送データの次のセグメントをデータラインD00からD31へ配置する849。

データの転送は、マスターよりスレーブに転送されるデータの全てがデータを記憶するために選択されたデバイスへ書き込まれるまで、上記のように継続する。転送データの全てが記憶デバイスに書き込まれると、マスターはアドレスライン、アドレス変更ライン、データライン、IACK\*ライン、LWORD\*ラインおよびDS0\*ラインを解放する852。マスターはハイからロウへ変化するDTACK\*の受信を持つ853。スレーブはDTACK\*をロウにし855、あらかじめ設定された時間の後、DTACK\*をハイとする857。ハイからロウへ変化するDTACK\*の受信に対応して、マスターはAS\*をハイとし、859、次にAS\*ラインを解放する861。

高速転送プロトコルを実現するために、従来の85mAトランジスタドライバーを、DTACK\*をドライブする目的で、VMEスレーブモジュールに従来のより使用されている48mAオープンコレクターの代わりに使用している。同時に、従来のVMEバスデータドライバーの代わりにSO型パッケージの84mAトランジスタドライバーが用いられている。後者の変更は両方のドライバーパッケージ自身のグラウンド端子のインダクタンスを減じ、従って、データ、DS0\*、およびDTACK\*間でスキューに寄与している“グラウンドバウンス”の影響を減じる効果がある。さらに、バスバックプレーンに沿った信号のリターンラインのインダクタンスは、多数のグラウンドピンを持つコネクタシステムを使用して低下でき、従って、信号のリターンおよび組み合わせたピンのインダクタンスも

最小にしている。そのようなコネクタシステムの例は、Teradyne Corporation 社製の“High Density Plus”コネクタモデルNo.420-8015-0000である。

付録C

パリティ-FIFO

パリティ-FIFO 240、280および270 (ネットワークコントローラ-110の) ならびに544および554 (記憶プロセッサ-114) は各々SICとして実装されている。全パリティ-FIFOは等価であり、電源投入時、または必要な特定の機能のオペレーションの期間に構成される。パリティ-FIFOは、異なるスピードのバス間のスピードのマッチングを行うため、パラレルSCS Iドライブのパリティの生成および訂正を実行するために設計される。

FIFOは、ポートAおよびポートBの二つの双方向のデータポートを有し、これらの間には38x84ビットのRAMバッファがある。ポートAは8ビットであり、ポートBは32ビットである。RAMバッファは二つの部分に分割され、RAM XおよびRAM Yと呼ばれ、各々が38x32ビットである。二つのポートはバッファの異なる二つの半分にアクセスし、利用可能な場合は、他の半分に交代してアクセスする。チップがパラレルパリティチップとして構成される場合は (例えば、SP114aのFIFO 544の一つ)、ポートBへの全アクセスがモニターされ、パリティはRAM XおよびRAM Yに交互に累積される。

チップはまたCPUインタフェースを有し、それは8ビットまたは16ビットである。16ビットのモードでは、ポートAのピンはCPUインタフェースの最上位データビットとして使用され、チップ中のFIFOのデータレジスタへ書き込みまたは読み込みがなされる場合に実際に使用されるのみである。

REQ、ACKハンドシェイクはポートAおよびポートBの両方でデータ転送に利用される。チップは、マスターモードでポートAのACK/RDY出力はチップがポートAでデータ転送可能であることを意味し、ポートAのREQ入力はスレーブが応答するという条件を指定しているということからは、チップはポートAでマスターまたはスレーブに構成可能である。スレーブモードでは、しかし、ポートAのREQ入力が、マスターがデータ転送が必要であることを指定して、データが利用できる時、チップはポートAのACK/RDYで応答する。チップはポートBのRWQをたて、データ転送の完了を示すためポート

をスレーブするの使われる。パリティ-sync 信号は、パリティ累積に因するデータの最後のワードがポートBにあることを示す目的で、パリティチップとして構成されたチップに指示するために使用される。ポートBのデータラインは、以下の条件が全て満足された場合のみ、Fifo チップによって駆動される。

- a. データ転送がポートAよりポートBへなされ、
- b. ポートB選択信号が高であり、
- c. ポートB出力イネーブル信号が高であり、そして、
- c. チップがパリティチップとして構成されていないか、または、

チップがパリティ修正モードであり、かつパリティ-Sync 信号が高である。リセット

この信号はチップ内の全レジスタをリセットし、全双方向ピンを高インピダンス状態におく。

動作説明

**通常動作** チップは通常、簡単なFIFOチップとして動作する。FIFOは、二つのRAMバッファを単純なピンポンモードで用いてシミュレートされる。データがポートBのFIFOへまたはFIFOよりバーストで入出力されることは、望ましいが必要な条件ではない。これは、ポートB選択信号をロウとし、ポートB Ack信号を出力することで実行される。データをポートBよりポートAへ転送するとき、データはまずRAM Xに書き込まれ、これがいっぱいになると、データ経路を、ポートBが、RAM Yへの書き込みを開始するように切り替えられる。一方、チップは、RAM Xの内容をポートAへ出力して、空にする。RAM Yが満杯で、RAM Xが空の場合は、データ経路は、ポートBが、RAM Xに再ロードし、ポートAがRAM Yをからにするように、再度切り替えられる。

**ポートAスレーブモード** これはデフォルトモードであり、チップはこの状態にリセットされる。このモードでは、チップは、SCS Iアダプター-542の一つのようなマスターが、データ転送のため、ポートA Requestをたてるのを待つ。もし、データがあれば、Fifo チップは、ポートA Ack/Rdyで応答する。

BのACKを待つので、ラップはポートBではマスターである。

信号仕様

ポートA 0-7、P

ポートAは8ビットのデータポートである。ポートA Pは、もし、使用されれば、このポートではオッドパリティである。

A Req, A Ack/Rdy

これらの二つの信号は、ポートAのハンドシェイクを制御するため、データ転送モードで使用される。

uP データ0-7、uPデータP、uPAdd 0-2、CS

これらの信号は、チップ内のプログラマブルレジスタのアドレス指定をするために、マイクロプロセッサにより使用される。データレジスタまたはチェックサムレジスタに書き込まれるとき、オッドパリティ信号uPデータPは、チェックのみされ、マイクロプロセッサパリティはイネーブルされる。Clik

クロック入力はチップタイミングを発生するの使用する。10-20MHzのレンジと思われる。

Read En, Write En

マイクロプロセッサアクセス時、CSが高のときは、これらの信号はマイクロプロセッサアクセスの方向を決定する。データ転送時、WDモードでは、これらの信号は、ポートA ACKと共に使用されるデータスレーブとなる。

ポートB 00-07、10-17、20-27、30-37、OP-3P

ポートBは32ビットデータポートである。各バイトに一つのオッドパリティがある。ポートB OPはビット00-07のパリティ、ポートB 1Pはビット10-17のパリティ、ポートB 2Pはビット20-27のパリティ、ポートB 3Pはビット30-37のパリティである。

B選択、B Req, B Ack, パリティ-Sync, B出力イネーブル

これらの信号はポートBでデータのハンドシェイクを制御するために、データ転送モードで使用される。ポートB ReqおよびポートB Ackは両方ポートB選択とゲートされる。ポートB Ack信号は、ポートBデータライン上でデータ

**ポートA WD モード** チップはWDモード、即ちウェスタンデジタルモードで定ように構成される。このモードでは、チップはポートAでスレーブモードとして構成されなければならない。これは、チップが読み込みイネーブルもしくは書き込みイネーブルで、ポートA Ack/Rdyで応答すると同時に、応答するという意味では、デフォルトスレーブモードとは異なっている。このモードはチップがウェスタンデジタル38C88 SCS IチップあるいはNCR 53C90 SCS Iチップとインタフェース可能になっている。

**ポートAマスターモード** チップがマスターとして構成される場合、データ転送可能時には、チップはポートA Ack/Rdyをたてる。この信号は、データがある場合は、ポートA Ack/Rdyで応答するDMAコントローラーのリクエスト入力へ接続することを想定している。DMAコントローラーのバースト可能なように、ポートA Ack/Rdy信号は、8または16バイトが転送される毎に否定される。

**ポートBパラレル書き込みモード** パラレル書き込みモードでは、チップは、ポートAよりポートBへのパラレル転送のため、パリティチップに構成される。このモードで、ポートB選択およびポートBリクエストがアサートされた場合、データは、ポートB Ack信号が受信される度、RAM XまたはRAM Yへ書き込まれる。最初の128バイトブロックで、データは選択されたRAMへ単にコピーされる。ポートBで駆動される次の128バイトは、最初の128バイトとの逆位相論理和がとられる。この手順は、パリティがこのチップに接続するように、全ドライブにたいして繰り返される。パリティ-Sync 信号は、128バイトの最後のブロックと共に、パラレルチップへアサートされる。これにより、チップは他のRAMへのアクセス切り替え可能となり、新規128バイトのパリティの累積を開始できる。

**ポートBパラレル読み込みモード-チェックデータ** このモードは、もし、全ドライブが読み込まれると、セットされ、パリティがチェックされる。この場合、データ転送コンフィギュレーションレジスタのパリティ修正ビットは、セットされていない。パリティチップはポートAで、まず、通常読み込みモードのように128バイトを読み、次に、ポートBリクエストをたてる。それに

特表平5-502525 (23)

より、この信号がアサートされ、チップは、ポートB Aclk信号をモニターし、ポートBのそのデータとその選択したRAM中のデータとの相違論理和を求める。パリティ-Syncは、最終ブロックの128バイトと共に、再度アサートされなければならない。このモードで、チップは、ポートBデータラインを駆動せず、その相違論理和の出力がゼロでないかチェックする。このとき、セットされているビットがあれば、パラレルパリティエラーのフラグがたつ。

**ポートBパラレル読み込みモード-データ修正** このモードは、データ転送コンフィギュレーションレジスタのパリティ-修正ビットをセットすることによって、セットされる。この場合、チップは、次に述べることを除いては、チェックモードとまったく同様に動作する：ポートB出力イネーブル、ポートB選択およびパリティ-Syncの真の時、データがポートBデータラインへ駆動され、ゼロのパラレルパリティチェックが実行されない。

**バイトスワップ** 通常モードでは、ポートBのビット00-07の第一バイト、ビット10-17が第二バイト、ビット20-27が第三バイト、およびビット30-37が各ワードの最終バイトである。これらのバイトの順序は、バイトアドレスビットが倒置されるように、コンフィギュレーションレジスタにバイトスワップビットを書き込むことによって、変更される。バイトの書き込みおよび読み込みの方法はCPUインタフェースが18ビットに構成されているか、8ビットに構成されているかに従う。以下の表は、ポートAリクエスト/応答のハンドシェイクを用い、データ転送の異なる可能性を示すバイト配列である。

CPU I/P	Invert Addr 1	Invert Addr 0	ポートB 00-07	ポートB 10-17	ポートB 20-27	ポートB 30-37
8	偽	偽	ポートA バイト0	ポートA バイト1	ポートA バイト2	ポートA バイト1
8	偽	真	ポートA バイト1	ポートA バイト0	ポートA バイト3	ポートA バイト2
8	真	偽	ポートA バイト2	ポートA バイト3	ポートA バイト0	ポートA バイト1
8	真	真	ポートA バイト3	ポートA バイト2	ポートA バイト1	ポートA バイト0
16	偽	偽	ポートA バイト0	uProc バイト0	ポートA バイト1	uProc バイト1
16	偽	真	uProc バイト0	ポートA バイト0	ポートA バイト1	uProc バイト1
16	真	偽	ポートA バイト1	uProc バイト1	ポートA バイト0	uProc バイト0
16	真	真	uProc バイト1	ポートA バイト1	uProc バイト0	ポートA バイト0

8ビットモードで、マイクロプロセッサポートよりFifoデータレジスタを積み込んだり、書き込んだりすることによって、Fifoがアクセスされる時、バイトは上の表に示される通りの順序であるが、ポートAの替わりに、uProcデータポートが用いられている。16ビットモードでは、上のテーブルが当てはまる。

**半増長転送**

データ転送が32ワード、または128バイトでない場合は、マイクロプロセッサは、全データが転送されるように、チップの内部レジスタを操作しなければならない。ポートA AclkおよびポートB Reqは通常、選択されたRAMの全32ワードが利用できるまで、アサートされない。これらの信号は、データ

転送ステータスレジスタの適切なRAMステータスビットに書き込みを実行し、強制的に駆動させられる。

半増長の転送が発生した場合、マイクロプロセッサは、どのレジスタを操作するより前に、両ポートが停止状態になるまで、待機しなければならない。そして、データ転送コントロールレジスタの、ポートAおよびポートB用のイネーブルデータ転送ビットの両方をリセットする。マイクロプロセッサはまた、それらのアドレスレジスタおよびRAMアクセスコントロールレジスタを読み込み、RAM XまたはRAM Yが、半増長データを保持していることを確認する。また、マイクロプロセッサは対応するアドレスレジスタを、18進数20に設定し、RAMをフルビットに強制し、アドレスを第一ワードにセットする。最後に、マイクロプロセッサは、チップに転送を完了させるため、イネーブルデータ転送ビットをセットする。

この時点で、Fifoチップは、RAMには全128バイトのデータがあると判断し、許可された場合は、128バイトを転送する。これら128バイトのあるものは、有効ではないという事実は、FIFOチップに、外部より認識されなければならない。

**プログラマブルレジスタ**

**データ転送コンフィギュレーションレジスタ (読み込み/書き込み)**

レジスタアドレス0。このレジスタはリセット信号でクリアされる。

ビット0 **WDモード**。データ転送がウェスタンディジタルWD 33C93Aプロトコルを使用していれば、セットする。そうでなければ、Adaptec 8250プロトコルが使用される。

ビット1 **パリティチップ**。このチップがポートBパリティを駆動するのなら、セットする。

ビット2 **パリティ修正モード**。パリティチップがポートBでパラレルパリティを修正するのなら、セットする。

ビット3 **18ビットの大きさのCPUインタフェース**。もし、セットされると、マイクロプロセッサデータビットはポ

ートAデータビットと結合され、16ビットポートが効率的に作り出される。マイクロプロセッサによる全てのアクセスならびにポートAリクエストおよび応答ハンドシェイクを使用して転送された全データは16ビットにより転送される。

ビット4 **反転ポートAバイトアドレス0**。ポートAバイトアドレスの最下位ビットを反転させるために、セットされる。

ビット5 **反転ポートAバイトアドレス1**。ポートAバイトアドレスの最上位ビットを反転させるために、セットされる。

ビット6 **チェックサムラップ**。アダーの最下位ビットへの桁上げする目的で、16ビットチェックサムアダーの桁上げをイネーブルするように、セットされる。

ビット7 **リセット**。このビットへ1を書き込むと、他のレジスタをリセットする。このビットは、最大2クロックサイクル後、自身をリセットして、通常0として読まれる。このビットへ書き込まれた後、最小4クロックサイクルの期間は、他のレジスタは書き込まれてはならない。

**データ転送コントロールレジスタ (読み込み/書き込み)**

レジスタアドレス1。このレジスタはリセット信号によるか、リセットビットへ書き込むことによって、クリアされる。

ビット0 **ポートAのイネーブルデータ転送**。ポートA Req/Aclkハンドシェイクをイネーブルするために、セットする。

ビット1 **ポートBのイネーブルデータ転送**。ポートB Req/Aclkハンドシェイクをイネーブルするために、セットする。

ビット2 **ポートAからポートB**。セットされると、データ転送はポートAからポートBへなされる。もし、リセットされると、データ転送はポートBからポートAへなされる。リクエストラインの故障を避けるために、このビットは、上記のイネーブルデータ転送ビット0または1と同時に変更さ



特表平5-502525 (24)

れてはならない。

ビット3 マイクロプロセッサパリティイネーブル。マイクロプロセッサインタフェースでパリティをチェックするには、セットする。Fifo データレジスタへ書き込むとき、Fifo データレジスタまたはチェックサムレジスタより読み込むとき、または18ビットでポートAリクエスト/応答の転送期間中に、パリティチェックはなされる。チップは、しかし、パリティを常時、再生成して、正しいパリティがRAMに書き込まれ、マイクロプロセッサインタフェースで読み取れるようにしている。

ビット4 ポートAパリティイネーブル。パリティがポートAでチェックされるときは、セットされる。Fifo データレジスタに18ビットモードでアクセスするとき、またはポートAリクエスト/応答の転送期間中に、チェックされる。チップは、しかし、パリティを常時、再生成して、正しいパリティがRAMに書き込まれ、ポートAインタフェースで読み取れるようにしている。

ビット5 ポートBパリティイネーブル。ポートBデータが有効なバイトパリティを持つば、セットされる。もし、セットされないと、RAMに書き込まれないとき、バイトパリティが内部的に生成される。バイトパリティは、ポートBより書き込まれるときは、チェックされないが、ポートBへ読み込まれるときは、いつもチェックされる。

ビット8 チェックサムイネーブル。18ビットのチェックサムレジスタへの書き込みをイネーブルするために、セットされる。このレジスタは、Fifo データレジスタへのアクセスおよびチェックサムレジスタへの全書き込みを含む、全RAMアクセスに関し、18ビットチェックサムを累積する。このビットは、チェックサムレジスタよりの

読み込みをする前に、リセットされなければならない。

ビット7 ポートAマスター。データ転送時、ポートAがマスターモードで動作するために、セットされる。

データステータスレジスタ (読みとりのみ)

レジスタアドレス2。このレジスタはリセット信号によるか、リセットビットへ書き込むことによって、クリアされる。

ビット0 RAM XまたはRAM Yのデータ。RAM X、RAM YまたはポートAバイトアドレスレジスタでいかなるビットが真でも、セットされる。

ビット1 マイクロプロセッサポートパリティエラー。マイクロプロセッサパリティイネーブルビットがセットされ、かつRAMアクセス時またはチェックサムレジスタへ18ビットモードで書き込み期間中に、パリティエラーが、マイクロプロセッサインタフェースに検出されると、セットされる。

ビット2 ポートAパリティエラー。ポートAパリティエラーがセットされ、かつ、いかなるRAMアクセス時、またチェックサムレジスタへの書き込み時に、パリティエラーがポートAインタフェースに検出されると、セットされる。

ビット3 ポートBパリティエラー。パリティチップとして構成されたチップが、パリティ修正モードになく、かつ、パリティSync 信号が真のとき、非ゼロの結果が検出されると、セットされる。また、データがポートBへ読み込まれ、双方方向のパワーで読み返されたデータが一致しない場合はいつでも、セットされる。

ビット4-7 ポートBバイト0-3パリティエラー。ポートBサイドでRAMより読み出されたデータが不良パリティを有するときは常に、セットされる。

RAMアクセスコントロールレジスタ (読み込み/書き込み)

レジスタアドレス3。このレジスタはリセット信号によるか、リセットビットへ書き込むことによって、クリアされる。データ転送コントロールレジスタのイネーブルデータ転送ビットは、このレジスタに書き込みをしようとする前に、リセットされていなければならない。そうでなければ、書き込みは無視される。

ビット0 ポートAバイトアドレス0。このビットは最下位のバイトアドレスビットである。これは、データ転送コンフィギュレーションレジスタでの、反転ビットによってなされるいかなる反転をバイパスして、直接読み取れる。

ビット1 ポートAバイトアドレス1。このビットは最上位のバイトアドレスビットである。これは、データ転送コンフィギュレーションレジスタでの、反転ビットによってなされるいかなる反転をバイパスして、直接読み取れる。

ビット2 ポートAからRAM Y。ポートAがRAM Yへアクセスしていれば、セットされ、RAM Xへアクセスしていれば、リセットされる。

ビット3 ポートBからRAM X。ポートBがRAM Yへアクセスしていれば、セットされ、RAM Xへアクセスしていれば、リセットされる。

ビット4 ロングバースト。もし、チップが、ポートAでマスターとしてデータを転送するように構成され、このビットがリセットされると、18ビットモードで8バイト、または4ワードで転送を完了する前に、チップはポートA Ack/Rdyを否定する。このビットがセットされると、ポートA Ack/Rdyは、18ビットモードで、18バイト毎または8ワード毎に否定される。

ビット5-7 不使用

RAM X アドレスレジスタ (読み込み/書き込み)

レジスタアドレス4。このレジスタはリセット信号によるか、リセットビットへ書き込むことによって、クリアされる。データ転送コントロールレジスタのイネーブルデータ転送ビットは、このレジスタに書き込みをしようとする前に、リセットされていなければならない。そうでなければ、書き込みは無視される。

ビット0-4 RAM X ワードアドレス  
 ビット5 RAM X フル  
 ビット6-7 不使用

RAM Y アドレスレジスタ (読み込み/書き込み)

レジスタアドレス5。このレジスタはリセット信号によるか、リセットビットへ書き込むことによって、クリアされる。データ転送コントロールレジスタのイネーブルデータ転送ビットは、このレジスタに書き込みをしようとする前に、リセットされていなければならない。そうでなければ、書き込みは無視される。

ビット0-4 RAM Y ワードアドレス  
 ビット5 RAM Y フル  
 ビット6-7 不使用

Fifo データレジスタ (読み込み/書き込み)

レジスタアドレス8。データ転送コントロールレジスタのイネーブルデータ転送ビットは、このレジスタに書き込みをしようとする前に、リセットされていなければならない。そうでなければ、書き込みは無視される。データ転送コントロールレジスタのポートAよりポートBビットは、このレジスタへ書き込みを前にセットされなければならない。そうでなければ、RAMコントロールは、無分されるが、データは、RAMに書き込まれない。一貫性をもたせるために、ポートAよりポートBは、このレジスタへ読み込む前に、リセットされる。ことが好ましい。

ビット0-7はFifo データである。マイクロプロセッサは、このレジスタへ読み込むかまたはこのレジスタへ書き込むかして、Fifo にアクセスをする。RAMコントロールレジスタは、あたかもアクセスがポートAを用いたが

特表平5-502525 (26)

加く、更新される。チップが16ビットのCPUインタフェースで構成されると、最上位バイトはポートA 0-7データラインを用い、各ポートAアクセスは、ポートA/バイトアドレスを2だけ増分する。

ポートAチェックサムレジスター (読み込み/書き込み)

レジスターアドレス7。このレジスターはリセット信号によるか、リセットビットへ書き込むことによって、クリアされる。

ビット0-7はチェックサムデータである。チップは、ポートAアクセスの全てにつき、16ビットのチェックサムを累積する。チップが16ビットCPUインタフェースと共に構成されていけば、最上位バイトはポートA 0-7のデータラインで読める。もし、データがこのレジスターに直接、書き込まれると、現在の内容に上書きされず、現在の内容に追加される。データ転送コントロールレジスターのチェックサムイネーブルビットは、このレジスターに書き込むためには、セットされていないければならず、読み込むためには、リセットされていないなければならない。

FIFOチップのプログラム

一般には、FIFOチップは、データ転送を可能とするため、データ転送コンフィギュレーションレジスターおよびコントロールレジスターに書き込むことによって、また完了ステータスをチェックするために、転送の終了時、データ転送ステータスレジスターを読み込むことによって、プログラムされる。通常、データ転送自身は、ポートAとポートBのハンドシェイクをイネーブルにして、実行される。この場合、データ転送自身は、他のいかなるマイクロプロセッサとの対話もなしで、実行されるべきである。いくつかのアプリケーションでは、ポートAハンドシェイクは、イネーブルされず、FIFOデータレジスターを繰り返し、書き込みまたは読み込むことによって、マイクロプロセッサがFIFOを讀み出すまたは空にすることが必要である。

FIFOチップはいかなるバイトカウントの情報も持っていないので、このチップ内のどのレジスターを読み込むことによって、どのデータ転送が完了したと示すことは不可能である。データ転送が完了したかどうかの決定は従ってこのチップ内の他の回路によってなされなければならない。

```
#define PORT-B.TU-RAM-Y(fifo)((fifo->ram-access-control)& 0x08)
/*****
  次のルーチンは、2つの重なるFIFOデータ転送の通過させて、FIFOデータ
  を放流している。
  config-data これはコンフィギュレーションレジスターへ書き込まれるデータ
  である。
  control-data これはデータ転送コントロールレジスターに書き込まれるデータ
  である。データ転送が、ポートAとポートBのハンドシェイク
  を用いて、自動的に実行される場合、両方のデータ転送イネー
  ブルビットはこのパラメーターでセットされなければならない。
  *****/
*****/
FIFO-Initiate-data-transfer(config-data, control-data)
unsigned char config-data, control-data;
{
    FIFO->config = config-data | FIFO_RESET; /*Set
  Configuration value & Reset*/
    FIFO->control = control-data & (~FIFO_PORT_ENABLES); /*Set
  everything but enables*/
    FIFO->control = control-data; /*Set data transfer
  enables*/
}
/*****
  次のルーチンは、データ転送の終了時、FIFOに読まれたどんな半端なバイト
  も強制的に転送する。それは、初めに、両ポートをディスエーブルし、次に、
  Ram Fullビットを強制的に駆動し、適切なポートを再イネーブルする。
  *****/
*****/
FIFO-force-odd-length-transfer()
{
```

次のC言語ルーチンは、バリエーションFIFOチップがどのようにプログラムされるかを示す。これらのルーチンは、次の規定に基づく：ポートAおよびマイクロプロセッサポートがシステムマイクロプロセッサと接続されていて、16ビットの大きさコードを戻しているが、ハードウェアはFIFOチップを長32ビットのレジスターとしてアドレス指定する。

```
struct FIFO_regs {
    unsigned char config_a1, a2, a3;
    unsigned char config_b1, b2, b3;
    unsigned char status, c1, c2, c3;
    unsigned char ram-access_control, d1, d2, d3;
    unsigned char ram-X.addr, e1, e2, e3;
    unsigned char ram-Y.addr, f1, f2, f3;
    unsigned long data;
    unsigned int checksum, hi;
};
#define FIFO((struct FIFO_regs*)FIFO-BASE-ADDRESS)
#define FIFO-RESET 0x80
#define FIFO-16-BITS 0x80
#define FIFO-CARRY-MSAP 0x40
#define FIFO-PORT-A-ENABLE 0x01
#define FIFO-PORT-B-ENABLE 0x02
#define FIFO-PORT-ENABLES 0x03
#define FIFO-PORT-A-TU-B 0x03
#define FIFO-CHECKSUM-ENABLE 0x03
#define FIFO-DATA-IN-RAM 0x01
#define FIFO-FORCE-RAM-FULL 0x20
#define PORT-A-TU-PORT-B(fifo)((fifo->control)& 0x04)
#define PORT-A-BYTE-ADDRESS(fifo)((fifo->ram-access-control)& 0x03)
#define PORT-A-TU-RAM-Y(fifo)((fifo->ram-access-control)& 0x04)

FIFO->control |= ~FIFO_PORT_ENABLES; /*Disable Ports A & B */
if(PORT-A-TU-PORT-B(FIFO)) {
    if(PORT-A-TU-RAM-Y(FIFO)) {
        FIFO->ram-Y.addr = FIFO-FORCE-RAM-FULL; /*
  Set RAM Y full*/
    }
    else FIFO->ram-X.addr = FIFO-FORCE-RAM-FULL; /*Set
  RAM X full*/
    FIFO->control |= FIFO-PORT-B-ENABLE; /*
  Re-Enable Port B*/
}
else {
    if(PORT-B-TU-RAM-Y(FIFO)) {
        FIFO->ram-Y.addr = FIFO-FORCE-RAM-FULL; /*
  Set RAM Y full*/
    }
    else FIFO->ram-X.addr = FIFO-FORCE-RAM-FULL; /*Set
  RAM X full*/
    FIFO->control |= FIFO-PORT-A-ENABLE; /*
  Re-Enable Port A*/
}
}
/*****
  次のルーチンは、データ転送終了時、FIFO中に残った半端バイトが残った
  場合、戻している。
  *****/
*****/
int FIFO-count-odd-bytes()
{
    int number-odd-bytes;
```

持表平5-502525 (26)

```

number-odd-bytes=0;
if(PIFO->status & PIFO-DATA-IN-RAM) {
  if(PORT-A -TO-PORT-B(PIFO)) {
    number-odd-bytes =
if(PORT-A-BYTE-ADDRESS(PIFO));
    if(PORT-A -TO-RAM-Y(PIFO))
      number-odd-bytes + = (PIFO->ram-Y.addr)*4;
4:
    else number-odd-bytes + = (PIFO->ram-Y.addr)*4;
  }
  else {
    if(PORT-B -TO-RAM-Y(PIFO))
      number-odd-bytes + = (PIFO->ram-Y.addr)*4;
    else number-odd-bytes + = (PIFO->ram-X.addr)*4;
  }
}
return(number-odd-bytes);
}

```

次のルーチンは、チップのマイクロプロセッサ-インタフェースをテストする。  
 まず、第一の8個のレジスタを書き込み読み込む。そして1、0およびアドレス  
 パターンをRAMに書き込み、読みとってチェックする。  
 テストは、故障したレジスタのアドレスを各ビットが表すビットシフトニフィ  
 カントエラーコードを返す。  
 ビット0=コンフィギュレーションレジスタ-故障  
 ビット1=コントロールレジスタ-故障  
 ビット2=ステータスレジスタ-故障  
 ビット3=RAMアクセスコントロールレジスタ-故障  
 ビット4=RAM Xアドレスレジスタ-故障

```

/*now test Ram data & checksum registers
test is throughout Ram & then test 0s*/
for(test-data -=1; test-data !=1; test-data++) /*test
for 1& 0s*/
PIFO->config = PIFO-RESET | PIFO-16-BITS;
PIFO->control = PIFO-PORT-A-TO-B;
for(i=0; i<RAM-DEPTH; i++) /*write data to RAM*/
PIFO->data = test-data;
PIFO->control = 0;
for(i=0; i<RAM-DEPTH; i++)
if(PIFO->data != test-data)error |= 1; /*read & check data*/
if(PIFO->checksum)error |= 0x80; /*checksum should = 0*/
}

/*now test Ram data with address pattern
uses a different pattern for every bytes*/
test-data= 0x00010203; /*address pattern starts*/
PIFO->config = PIFO-RESET | PIFO-16-BITS |
PIFO-CARRY-WARP;
PIFO->control = PIFO-PORT-A-TO-B |
PIFO-CHECKSUM-ENABLE;
for(i=0; i<RAM-DEPTH; i++) {
  PIFO->data = test-data; /*write address pattern*/
  test-data += 0x04040404;
}
test-data= 0x00010203; /*address pattern starts*/
PIFO->control = PIFO-CHECKSUM-ENABLE;
for(i=0; i<RAM-DEPTH; i++) {
if(PIFO->status & PIFO-DATA-IN-RAM)
error |= 0x04; /*should be data in ram*/
}

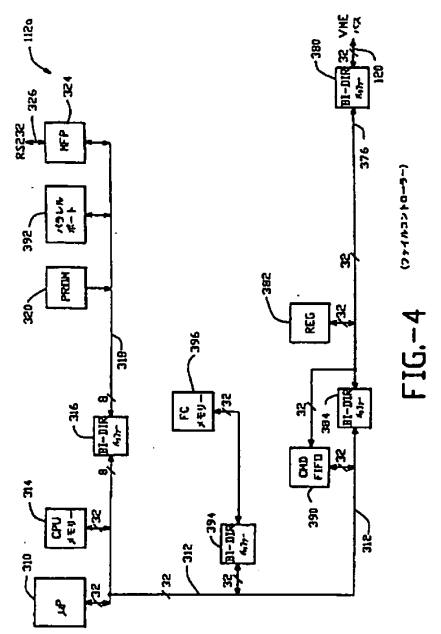
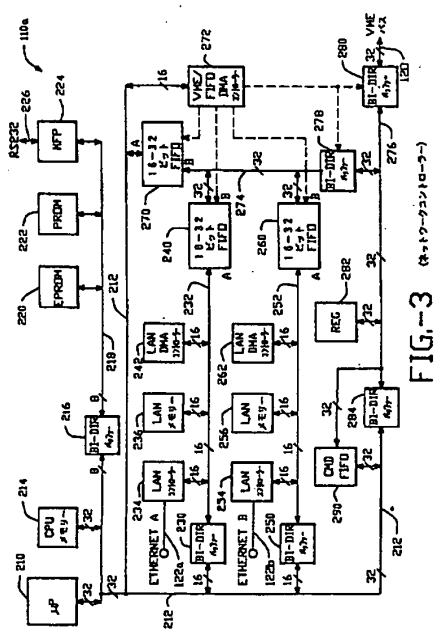
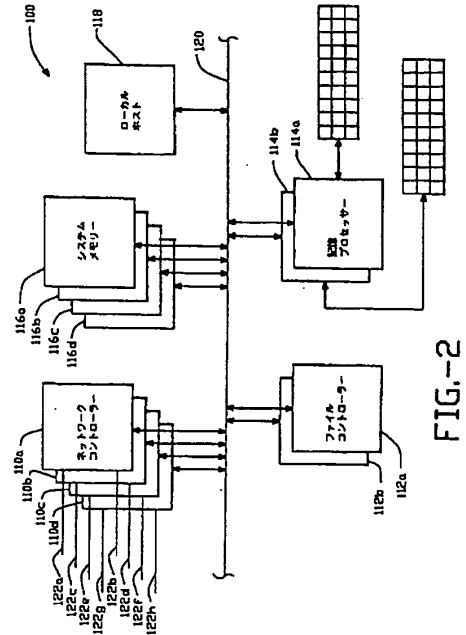
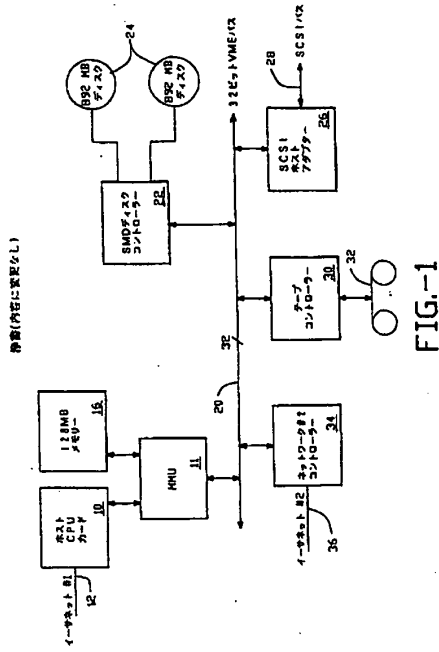
```

```

ビット5=RAM Yアドレスレジスタ-故障
ビット6=データレジスタ-故障
ビット7=チェックサムレジスタ-故障
*****
#define RAM-DEPTH 64 /*number of long words in Pifo RAM*/
reg - expected_data[6] = {0x7F,0xFF,0x00,0x1F,0x3F,0x5F};
char PIFO-sprocessor-interface-test[]
{
  unsigned long test_data;
  char* register_addr;
  int i;
  char j; error;
  PIFO->config = PIFO-RESET; /*reset the chip*/
  error=0;
  register_addr =(char*)PIFO;
  j=1;
  /*first test registers 0 thru 5*/
  for (i=0; i<8;i++) {
    *register_addr=0xFF; /*write test data*/
    if(*register_addr!=reg-expected_data[i])error |=j;
    *register_addr=0; /*write 0s to registers*/
    if(*register_addr)error |=j;
    *register_addr=0xFF; /*write test data again*/
    if(*register_addr!=reg-expected_data[i])error |=j;
    PIFO->config = PIFO-RESET; /*reset the chip*/
    if(*register_addr)error |=j; /*register should be 0*/
    *register_addr++; /*go to next data registers*/
    j<<=1;
  }

  if(PIFO->data != test-data)error |= 1; /*read & check address
pattern*/
  test-data += 0x04040404;
}
if(PIFO->checksum !=0x0102) error |=0x80; /*test checksum of
address pattern*/
PIFO->config=PIFO-RESET | PIFO-16-BITS; /*inhibit carry wrap*/
PIFO->checksum = 0xFFFE; /*write adds to checksum */
if(PIFO->checksum)error |=0x80; /*checksum should be 0*/
return(error);
}

```



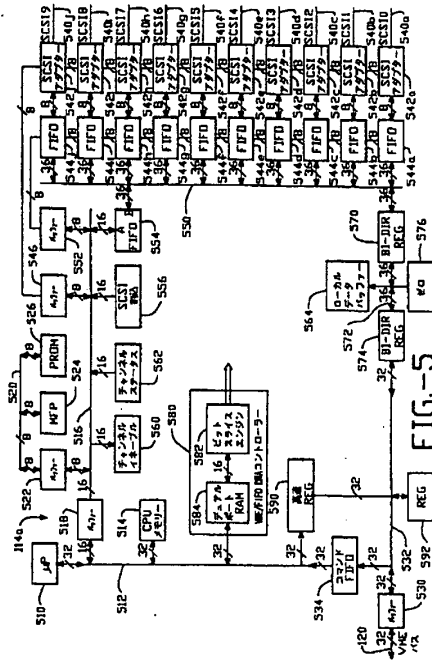


FIG. 5

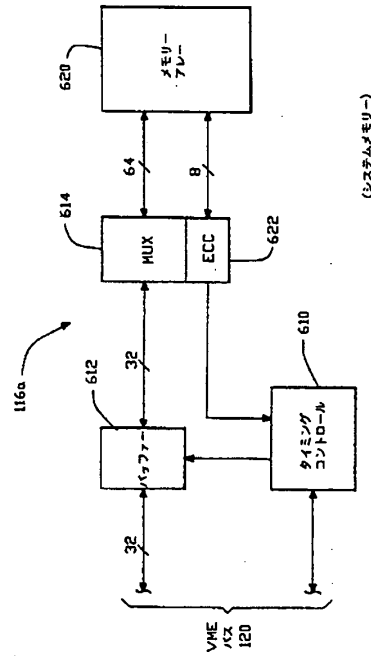


FIG. 6

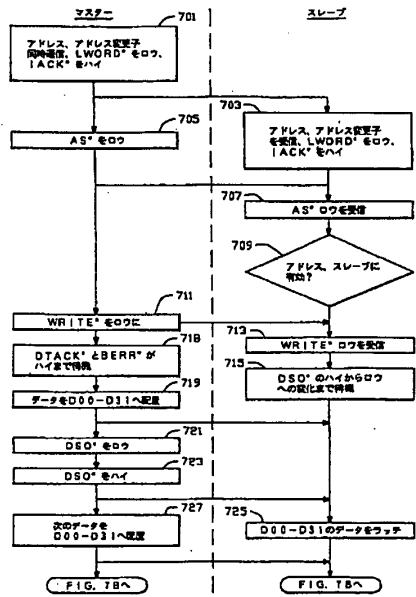


FIG. 7A

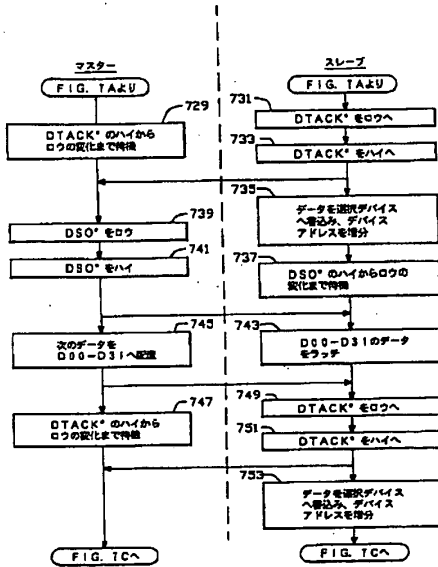


FIG. 7B

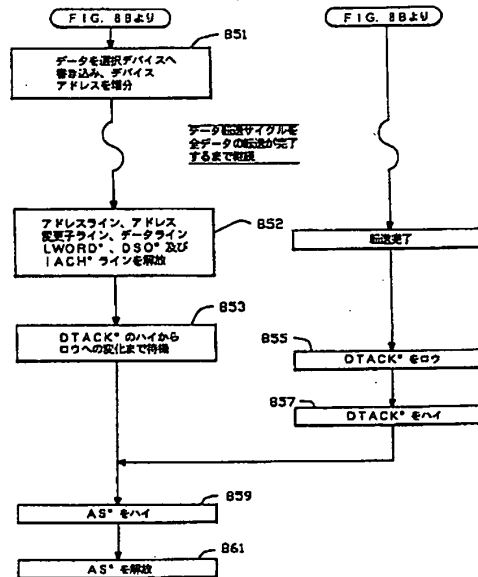
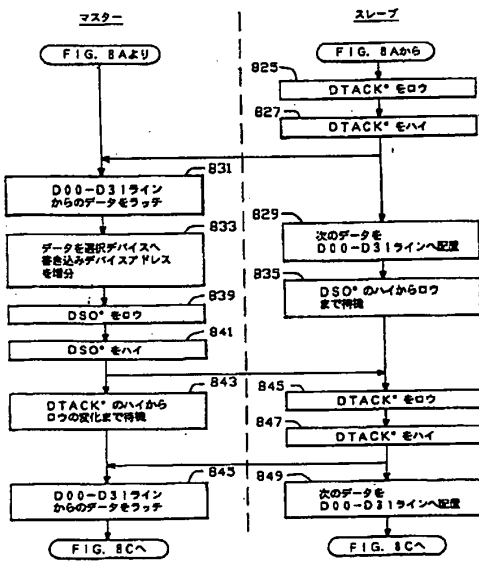
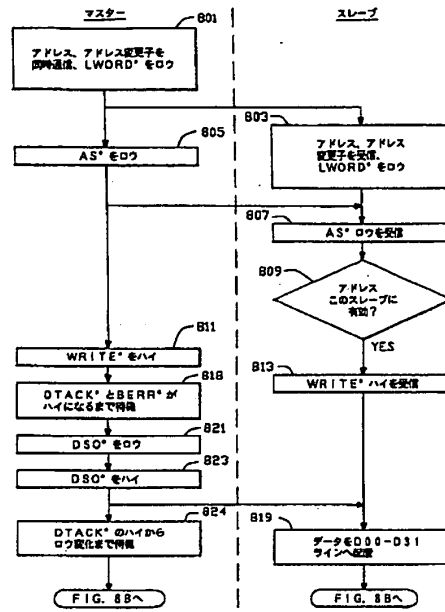
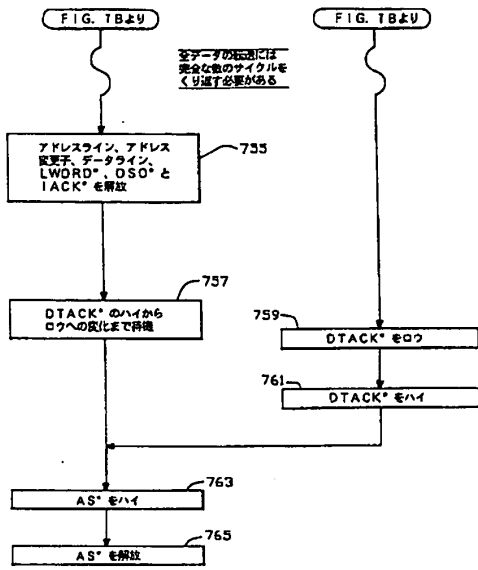


FIG.-8B

FIG.-8C



(19)日本国特許庁(JP)

(12)公開特許公報 (A)

(11)特許出願公開番号

特開平5-181609

(43)公開日 平成5年(1993)7月23日

(51)Int. Cl.<sup>5</sup>  
G 0 6 F 3/06

識別記号 庁内整理番号  
3 0 1 Z 7165-5 B

F I

技術表示箇所

審査請求 未請求 請求項の数 1

(全 4 頁)

(21)出願番号 特願平4-333

(22)出願日 平成4年(1992)1月6日

(71)出願人 000004237

日本電気株式会社  
東京都港区芝五丁目7番1号

(72)発明者 平井 秀生

東京都港区芝五丁目7番1号 日本電気株式会社内

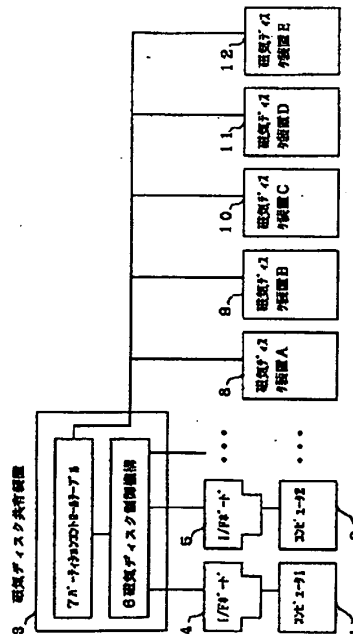
(74)代理人 弁理士 若林 忠

(54)【発明の名称】 パーソナルコンピュータシステム

(57)【要約】

【目的】 複数の磁気ディスク装置を複数のパーソナルコンピュータで共用できるパーソナルコンピュータシステムを提供する。

【構成】 複数の磁気ディスク装置8~12を、それらの全記憶領域を記憶領域とする1個の仮想磁気ディスク装置とみなして制御する磁気ディスク制御機構6と、仮想磁気ディスク装置の記憶領域のパーティションごとに指定する各パーソナルコンピュータ1, 2, ...のアクセス権を管理するパーティション・コントロール・テーブル7とを備えた磁気ディスク共有装置3を有し、各パーソナルコンピュータはそれぞれのアクセス権にしたがって仮想磁気ディスク装置にアクセスする。





## 【特許請求の範囲】

【請求項1】 複数のパーソナルコンピュータと複数の磁気ディスク装置を含むパーソナルコンピュータシステムにおいて、

前記複数の磁気ディスク装置を、該複数の磁気ディスク装置の全記憶領域をその記憶領域とする1個の仮想磁気ディスク装置とみなして制御する磁気ディスク装置の制御手段と、当該仮想磁気ディスク装置の記憶領域における前記複数のパーソナルコンピュータそれぞれの利用可能な権利を管理するセキュリティ管理手段とを備えた磁気ディスク共有装置を有し、

前記複数のパーソナルコンピュータはそれぞれ利用可能な前記権利にしたがって前記仮想磁気ディスク装置にアクセスすることを特徴とするパーソナルコンピュータシステム。

## 【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、複数のパーソナルコンピュータと複数の磁気ディスク装置を含むパーソナルコンピュータシステムに関する。

【0002】

【従来の技術】従来、パーソナルコンピュータシステムにおける磁気ディスク制御装置は1個の論理磁気ディスク装置に対して1個の物理磁気ディスク装置しか対応できず、上位ソフトウェア（オペレーティングシステム）の磁気ディスク管理も同様となっている。

【0003】また、磁気ディスク装置を複数のパーソナルコンピュータで共用できなかつた。

【0004】

【発明が解決しようとする課題】上述した従来の磁気ディスク制御装置は、1個の論理磁気ディスク装置に1個の物理磁気ディスク装置しか対応できない仕様となっているため、1個の磁気ディスク装置の容量を超えるデータを扱うことができないという欠点があり、また磁気ディスク装置を複数のパーソナルコンピュータによって共用することができないという欠点があった。

【0005】本発明の目的は、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域として個々のパーソナルコンピュータからアクセスすることにより、一台の磁気ディスク装置の容量を超える大きさのデータを扱うことを可能にするとともに、個々のパーソナルコンピュータのアクセス権を管理しつつ、複数のマイクロコンピュータによって複数の磁気ディスク装置を共用することのできる磁気ディスク制御装置を有するマイクロコンピュータシステムを提供することである。

【0006】

【課題を解決するための手段】本発明のパーソナルコンピュータシステムは、複数の磁気ディスク装置を、それら複数の磁気ディスク装置の全記憶領域をその記憶領域

とする1個の仮想磁気ディスク装置とみなして制御する磁気ディスク装置の制御手段と、その仮想磁気ディスク装置の記憶領域における複数のパーソナルコンピュータそれぞれの利用可能な権利を管理するセキュリティ管理手段とを備えた磁気ディスク共有装置を有し、複数のパーソナルコンピュータはそれぞれ利用可能な権利にしたがって仮想磁気ディスク装置にアクセスする。

【0007】

【作用】個々のパーソナルコンピュータにとり、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域であるかのようにアクセスすることを可能とし、その1個の仮想磁気ディスク装置の記憶領域に対する複数のパーソナルコンピュータそれぞれの利用可能な権利は個々のマイクロコンピュータごとに指定され管理される。

【0008】

【実施例】次に、本発明の実施例について図面を参照して説明する。

【0009】図1は本発明の、磁気ディスク共用装置を含むパーソナルコンピュータシステムの一実施例の構成を示すブロック図である。

【0010】図1において、パーソナルコンピュータ本体1, 2, ...はインタフェースボード4, 5, ...を介して磁気ディスク共用装置3に接続されている。また、磁気ディスク共用装置3には磁気ディスク装置8~12が接続されている。磁気ディスク共用装置3は、磁気ディスク制御機構6とパーティション・コントロール・テーブル7から構成されている。

【0011】パーソナルコンピュータ1, 2, ...から磁気ディスク装置8~12へのアクセス要求は、磁気ディスク用インタフェースボード4, 5, ...を通じ磁気ディスク制御機構6に通知され、磁気ディスク制御機構6において磁気ディスク装置8~12にまたがる仮想的な磁気ディスク装置に対するアクセス要求に変換される。以上の処理によりパーソナルコンピュータ本体からは、磁気ディスク装置8~12を、磁気ディスク装置8~12の全記憶領域を自らの記録領域とする仮想化された一つの磁気ディスク装置として扱うことが可能となる。

【0012】セキュリティ管理は、上述の仮想磁気ディスク装置の記憶領域を区分けし、区分けされた各部分（パーティションと云う）に、個々のパーソナルコンピュータごとの利用可能な権利を設定し、不正なアクセスを防ぐためのもので、パーティション・コントロール・テーブル7を作成して行われる。パーティションへのアクセス権には、R（読み出し）、W（書き込み）、C（作成）、D（消去）、X（実行）がある。

【0013】図2はパーティション・コントロール・テーブルの例である。パーソナルコンピュータ1は、パーティション1に対し読み出し、書き込み、作成、実行が

可能であり、パーティション2に対し読み出し、書き込みが可能であり、パーティション3に対し読み出しが可能である。パーソナルコンピュータ2は、パーティション1に対し読み出し、書き込み、作成、実行が可能であり、パーティション3に対し読み出しが可能である。パーソナルコンピュータ3は、パーティション1に対し読み出し、書き込み、作成、実行が可能であり、パーティション2に対し読み出しが可能である。上記セキュリティ管理手段により、パーソナルコンピュータからの利用の許されていない不正なアクセスを防止することが可能となる。

【0014】

【発明の効果】以上説明したように、本発明の磁気ディスク共用装置を有するパーソナルコンピュータシステムは、複数の磁気ディスク装置の全記憶領域をあたかも1個の磁気ディスク装置の記憶領域として個々のパーソナルコンピュータからアクセスし、かつ各パーソナルコンピュータのアクセス権を管理することにより、1個の磁気ディスク装置の記憶容量を超える大きさのデータを扱

うことを可能にするとともに、複数のマイクロコンピュータにより複数の磁気ディスク装置を共用することを可能にし、データを個々のパーソナルコンピュータで保管することなく一括して管理することができる効果がある。

【図面の簡単な説明】

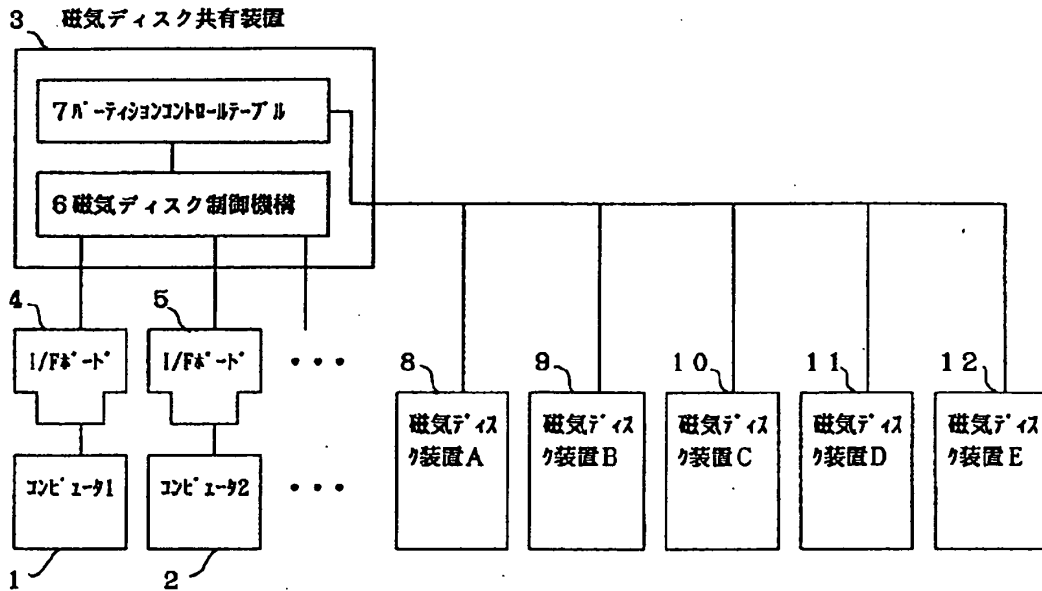
【図1】本発明のパーソナルコンピュータシステムの一実施例である。

【図2】パーティション・コントロール・テーブルの例である。

【符号の説明】

- 1 パーソナルコンピュータ本体1
- 2 パーソナルコンピュータ本体2
- 3 磁気ディスク共有装置
- 4~5 インタフェースボード
- 6 磁気ディスク制御機構
- 7 パーティション・コントロール・テーブル
- 8~12 磁気ディスク装置

【図1】



【図2】

ノード名	コンピュタ名	アクセス権
ノード1	ノード1-1	RWCX
	ノード1-2	RWCX
	ノード1-3	RWCX
ノード2	ノード2-1	RW
	ノード2-3	R
ノード3	ノード3-1	R
	ノード3-2	R
ノードn	⋮	⋮

(R:読みだし可 W:書き込み可 C:作成可 X:実行可)

(19) 日本国特許庁(JP)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平7-20994

(43) 公開日 平成7年(1995)1月24日

(51) Int. Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F	3/06	3 0 1 B		
	12/08	3 2 0	7608-5 B	
	13/12	3 3 0 T	8133-5 B	

審査請求 未請求 請求項の数 9 O L (全21頁)

(21) 出願番号	特願平5-162021	(71) 出願人	000005108 株式会社日立製作所 東京都千代田区神田駿河台四丁目6番地
(22) 出願日	平成5年(1993)6月30日	(72) 発明者	二宮 龍也 神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内
		(72) 発明者	増崎 秀文 神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内
		(72) 発明者	黒沢 弘幸 神奈川県小田原市国府津2880番地 株式会社日立製作所ストレージシステム事業部内
		(74) 代理人	弁理士 武 顕次郎

最終頁に続く

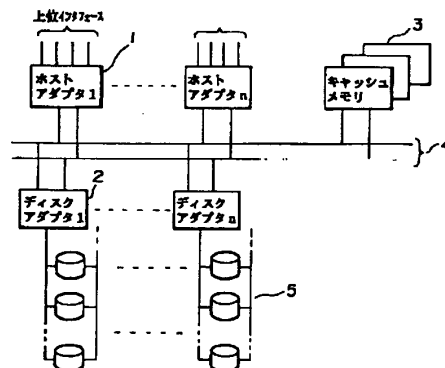
(54) 【発明の名称】 記憶システム

(57) 【要約】

【目的】 大形計算機の記憶システムで、システム規模を容易に拡張変更でき、システムの縮退及び活線挿抜による保守を可能とする。

【構成】 上位CPUと接続される複数のホストアダプタ(上位側インタフェース)1と、アレイドスク5と接続される複数のディスクアダプタ(記憶装置側インタフェース)2と、これらのアダプタに共用される一時記憶用キャッシュメモリ3とは、これらアダプタ及びキャッシュメモリに共用される共通バス4上に挿抜自在に取り付けられる。規模を拡大するには、必要な数だけこれらアダプタ1, 2及びキャッシュメモリ3を付加するだけでよい。アダプタ1, 2, キャッシュメモリ及び共通バスは二重化され、障害時の縮退運転を可能とし、また各アダプタ及びキャッシュメモリと共通バスとの結合部は、活線挿抜可能としシステム無停止で保守点検部品交換を可能とする。

【図1】



## 【特許請求の範囲】

【請求項 1】 上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置及び前記複数の記憶装置側接続論理装置間で転送されるデータを一時記憶するキャッシュメモリ装置とを有する記憶システムにおいて、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、これらの装置に共用される共通バスにより相互に接続されるように構成したことを特徴とする記憶システム。

【請求項 2】 前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、いずれもモジュールで構成し、前記モジュールは、それぞれ、前記共通バスに対し挿抜自在に取付けられるように構成したことを特徴とする請求項 1 記載の記憶システム。

【請求項 3】 前記共通バスは、プラッタ上に配設され、前記上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記プラッタに対し挿抜自在に取付けられるように構成したことを特徴とする請求項 1 または 2 記載の記憶システム。

【請求項 4】 前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記共通バスは、いずれも少なくとも二重化されており、前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記共通バスの一方により縮退運転が可能となるように構成したことを特徴とする請求項 1 ないし 3 のいずれか 1 記載の記憶システム。

【請求項 5】 前記二重化された上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置は、いずれも活線挿抜ができるように構成したことを特徴とする請求項 4 記載の記憶システム。

【請求項 6】 前記記憶装置は、二重化された電源部を備えたことを特徴とする請求項 1 ないし 6 のいずれか 1 記載の記憶システム。

【請求項 7】 前記記憶装置は、複数の小形記憶装置を組み合わせたアレイ記憶装置で構成したことを特徴とする請求項 1 ないし 3 のいずれか 1 記載の記憶システム。

【請求項 8】 前記キャッシュメモリ装置は、キャッシュメモリ本体を持ち、前記共通バスに直接取り付けられるキャッシュメモリモジュールと、キャッシュメモリを持つ増設用のキャッシュユニットとを有しており、前記キャッシュユニットは、前記共通バスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して接続されるように構成したことを特徴とする請求項 1～7 のいずれか 1 記載の記憶システム。

【請求項 9】 前記上位側接続論理装置及び前記記憶装置側接続論理装置は、それぞれ、二重化されたマイクロプロセッサを有し、両マイクロプロセッサによりデータの比較チェックを行なうように構成したことを特徴とする請求項 1 ないし 8 のいずれか 1 記載の記憶システム。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】本発明は、大形計算機システムやネットワークシステム等に接続される磁気ディスク装置、磁気テープ装置、半導体記憶装置、または光ディスク装置等の記憶装置を制御する記憶制御装置を含む記憶システムに係り、特に、システムの拡張性が高く縮退運転や活線挿抜対応の可能な記憶システムに関する。

## 【0002】

【従来の技術】従来、大形計算機に接続される記憶システムとして、例えば特開昭 61-43742 号公報に記載されているように、上位装置 (CPU) に対するインタフェース (ホストアダプタ)、キャッシュメモリ、及び磁気ディスク装置等の記憶装置に対するインタフェース (ディスクアダプタ) の相互間をホットライン (専用線) で接続しているものが知られている。

【0003】図 20 は、従来の記憶システムの構成の概要を示す図である。同図において、201-1～201-n はそれぞれ複数の上位ホスト (CPU) に接続されるホストアダプタ (対上位論理モジュール)、202-1～202-n は、共有の大形ディスク装置 205 に接続されるディスクアダプタ (記憶媒体接続用論理モジュール)、203 は、複数のホストアダプタに共有のキャッシュメモリ、206 は同様に共有の管理メモリである。従来装置では、各ホストアダプタ 201-1～201-n とキャッシュメモリ 203 の間、キャッシュメモリ 203 と各ディスクアダプタ 202-1～202-n の間、各ホストアダプタ 201-1～201-n と管理メモリ 206 の間、並びに管理メモリ 206 と各ディスクアダプタ 201-2～201-n の間は、それぞれ別々のホットライン 207-1～207-n 及び 208-1～208-n によって接続されている。また、これらのホストアダプタ及びディスクアダプタの監視及び保守を行なう保守用プロセッサ (SVP、図示せず) も各々のホストアダプタ及びディスクアダプタにそれぞれ専用線を介して接続されている。

## 【0004】

【発明が解決しようとする課題】上記従来技術では、上位装置に対するホストアダプタ (対上位接続論理モジュール) と、記憶装置に対するディスクアダプタ (対記憶媒体接続論理モジュール) と、キャッシュメモリ (キャッシュメモリモジュール) との各間がホットラインで接続されているため、装置構成が複雑になると共に、ホストアダプタ、キャッシュメモリ、ディスクアダプタ、ディスク装置等、装置としての拡張性に乏しくいわゆるス

ケーラブル(拡張及び縮小自在)なシステム構成が得られなかった。また、システムを多重化することにより障害発生時等に縮退運転(2台のうち1台を停止し他の1台だけで運転するなど)や活線挿抜対応(システムを動作したままで基板や回路の部品等を挿しかえるなど)を可能とすることがなにも配慮されておらず、このため、障害発生時の部品交換やシステムの制御プログラムをグレードアップするときには、システムを一時停止し対応しなければならない等の問題があった。

【0005】従って、本発明の目的は、上記従来技術の問題点を解決し、コモンバス方式を採用することにより、システム構成(規模)に応じてホストアダプタ、記憶装置アダプタ等の各論理モジュールやキャッシュメモリ及び記憶媒体を接続することでスケラブルなシステムを実現することができるようにすると共に、各論理モジュール、記憶媒体及びコモンバスの多重化により、縮退運転と各論理モジュール及び記憶媒体の活線挿抜対応とを可能とし、無停止で保守することができる記憶システムを提供することにある。

【0006】

【課題を解決するための手段】上記目的を達成するため、本発明は、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、前記複数の上位側接続論理装置及び前記複数の記憶装置側接続論理装置間で転送されるデータを一時記憶するキャッシュメモリ装置とを有する記憶システムにおいて、前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、これらの装置に共用されるコモンバスにより相互に接続されるように構成する。

【0007】前記複数の上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置は、いずれもモジュールで構成し、前記モジュールは、それぞれ、前記コモンバスに対し挿抜自在に取付けられるように構成する。

【0008】前記コモンバスは、プラッタ上に配設され、前記上位側接続論理装置、前記複数の記憶装置側接続論理装置、及び前記キャッシュメモリ装置を構成するモジュールの各々は、前記プラッタに対し挿抜自在に取付けられるように構成する。

【0009】前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記コモンバスは、いずれも少なくとも二重化されており、前記上位側接続論理装置、前記記憶装置側接続論理装置、前記キャッシュメモリ装置、及び前記コモンバスの一方により縮退運転が可能となるように構成する。

【0010】前記二重化された上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置は、いずれも活線挿抜ができるように構成する。

【0011】前記記憶装置についても、同様に二重化された電源部を備えることができる。

【0012】前記記憶装置は、複数の小形記憶装置を組み合わせたアレイ記憶装置で構成することができる。

【0013】前記キャッシュメモリ装置は、キャッシュメモリ本体を持ち、前記コモンバスに直接取り付けられるキャッシュメモリモジュールと、キャッシュメモリを持つ増設用のキャッシュユニットとを有しており、前記キャッシュユニットは、前記コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して接続されるように構成することができる。

【0014】前記上位側接続論理装置及び前記記憶装置側接続論理装置は、それぞれ、二重化されたマイクロプロセッサを有し、両マイクロプロセッサによりデータの比較チェックを行なうように構成することができる。

【0015】なお、コモンバス上には、上記上位側接続論理モジュールとは別の形式の上位側インタフェースや、上記記憶装置側接続論理モジュールとは別の形式の記憶装置側インタフェースを置き換えたり増設したりすることもできる。

【0016】

【作用】上記構成に基づく作用を説明する。

【0017】本発明によれば、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、これらの装置間で転送されるデータを一時記憶するキャッシュメモリ装置(複数の上位側接続論理装置及び複数の記憶装置側接続論理装置に共有されるキャッシュメモリ装置)とを有する記憶システムにおいて、前記複数の上位側接続論理装置、複数の記憶側接続論理装置、及びキャッシュメモリ装置は、これらの装置に共有されるコモンバスにより相互に接続されるように構成したので、上位側接続論理装置と記憶装置側接続論理装置とキャッシュメモリの増設または変更は、単にこれらをコモンバス上に追加しまたは変更して行くだけでよく、増設によるアップグレードが容易に達成できスケラブルなシステム構成をすることができる。

【0018】また、これらの上位側接続論理装置、記憶装置側接続論理装置及びキャッシュメモリ装置は、モジュール化されて、コモンバスの配設されたプラッタに挿抜(着脱)自在に取り付けるようにしたので、これらの装置の必要な数量の増設作業も簡単である。

【0019】また、上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置、及びこれらの間を接続するコモンバスは、二重化され、2系統に分けて配線されているので、これらの装置の一方に障害が発生したときでも、他方の装置を用いて縮退運転が可能である。なお、障害発生時に縮退運転状況を示す情報は、共有メモリに書き込まれる。

【0020】この場合、上位側接続論理装置、記憶装置側接続論理装置、及びキャッシュメモリ装置は、いずれも活線挿抜対応のコネクタ部を具備しているため、システムを停止することなく保守点検を行なって故障部品の交換を行ったり、増設用の部品を追加したりすることが可能である。

【0021】電源部も二重化され、それにより無停電電源装置を実現する。

【0022】記憶装置は、複数の小形記憶装置を組み合わせたアレイ形とされ、これにより従来の大形ディスク装置1台を用いたものに比べてアクセスタイムを短縮できる。

【0023】キャッシュメモリ装置は、コモンバスに直接取り付けられるキャッシュメモリモジュール（キャッシュメモリパッケージ）と、増設用のキャッシュユニットとで構成され、増設用のキャッシュユニットは、コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して必要数接続されるようになっているので、簡単に増減することができる。

【0024】異常により、高信頼性の記憶システムを得ることができる。

【0025】

【実施例】以下に、本発明の実施例を図面の図1から図18により説明する。

【0026】図1は本発明の概念図を示す。図1により、本実施例の概要を説明する。

【0027】1は、対上位CPU（ホスト）接続用論理モジュールであるホストアダプタ部、2は、対記憶媒体接続用論理モジュールであるディスクアダプタ部、3は、両モジュール間で転送されるデータを一時記憶するキャッシュメモリパッケージ（キャッシュメモリモジュール）、4はホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3の間のデータ転送制御を司るコモンバス、5は、縦横にアレイ状に配置した記憶媒体である磁気ディスク群（以下「アレイディスク」という）である。ホストアダプタ1は、上位インタフェース側のデータ形式及びアドレス形式を記憶媒体インタフェース用のデータ形式及びアドレス形式に変換する手段と、これらを制御管理する二重化したマイクロプロセッサとを有している。ディスクアダプタ2は、記憶媒体へデータを格納するためのアドレス演算機能と、記憶データ保証用冗長データの生成機能と、記憶媒体構成情報を認識する機能と、これらを制御管理するマイクロプロセッサとを有している。

【0028】図1において、上位装置（CPU）から送られてきた書き込みデータは、ホストアダプタ1からコモンバス4を介して一度キャッシュメモリパッケージ3に書き込むことにより上位に終了報告を行い、その後の空き時間でキャッシュメモリパッケージ3からディスクアダプタ2を経由してアレイディスク5に書き込む。

【0029】また、上位装置からのデータ読み出し命令に対しては、キャッシュメモリパッケージ3上にデータが存在する場合はアレイディスク5からは読み出さず、キャッシュメモリパッケージ3上のデータを上位装置に転送する。一方キャッシュメモリパッケージ3上にデータが存在しない場合は、アレイディスク5からディスクアダプタ2によりコモンバス4を経由して一度キャッシュメモリパッケージ3に書き込まれた後同様にホストアダプタ1を経由して上位装置へ転送する。

【0030】コモンバス4上のホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3各々はその接続数を任意に変えることができる。ホストアダプタ1の実装数を変えれば対上位接続バス数が増加し、上位ホストに対するデータ転送能力を高めることができる。ディスクアダプタ2の実装数を変えれば記憶媒体に対する接続バス数が増加し、記憶媒体に対するデータの書き込み/読み出しの転送能力を高めることができる。また、同時に記憶媒体の数も増加することができる。キャッシュメモリパッケージ3の実装数を変えればデータの一時格納場所であるキャッシュメモリの容量が増加し、記憶媒体の総容量に対するキャッシュメモリの容量の比率を高めることができるので、対上位装置からアクセスするデータがキャッシュメモリ上に存在する確率（以下「キャッシュヒット率」という）を高める等スケラブルな装置構成を実現できる。

【0031】図2は、図1の概念図の詳細な構成図を示したものである。図2は、図1の複数台のホストアダプタ及び複数台のディスクアダプタのうち、それぞれ1台だけを示し、他は図示を省略している。

【0032】ホストアダプタ1において、6はホストインタフェースの光信号を電気信号に変換する信号変換部、7は上位データフォーマットをアレイディスク5用フォーマットに変換するフォーマット変換部である。8はコモンバス4とのデータの授受を司るデータ転送制御部で、内部にパケット転送単位のデータを格納する記憶バッファを内蔵している。9は活線挿抜対応可能な小振幅電流駆動形バスタライバ（以下「BTL」という）である。

【0033】ホストからのデータ転送要求は10のマイクロプロセッサ（以下「MP」という）に引継がれ、ホストアダプタ1内のデータ転送制御は当MP10の管理下で行われる。

【0034】MP10はMP内の障害発生を検出するなど高信頼性を確保するために二重化されており、11のチェッカ部で同じ動作をする二重化されたMP10とMP10'を比較チェックしている。

【0035】12はMP10の制御プログラムを格納するブートデバイスで、このブートデバイス12には書き換え可能な大容量フラッシュメモリを採用しており、またMP10は必要に応じて13のローカルメモリに制御

プログラムをコピーして使用することにより、MP10のメモリアクセス時間の高速化を実現しており、図中破線で囲まれた部分29がチャンネルアダプタモジュールであり、ホストアダプタ1には当モジュール29が2回路搭載してある。

【0036】ディスクアダプタ2において、14はアレイドディスクに書き込むデータをセクタ単位に格納するバッファメモリ、15はバッファメモリ14の制御及びデータ転送制御を行なうデータ制御バッファ部、16はアレイドディスク5に書き込むデータを保証するための冗長データ生成部、17はアレイドディスク5（ターゲット）に対するイニシエータ（SCSIのマスタ側インタフェース）である。

【0037】またディスクアダプタ2内のデータ転送制御は、ホストアダプタ1と同じ構成をとるMP周辺部（MP10、MP10'、チェッカ11、ブートデバイス12、ローカルメモリ13からなりディスクアダプタ用の制御プログラムを搭載する）の管理下で行なわれる。

【0038】アレイドディスク5は、図2では4つのディスク（ターゲット）しか示していないが、実際には1台のディスクアダプタ2に対し例えば4（横）×4（縦）～4（横）×7（縦）つのディスクで構成される。横列はECCグループ（Error Correction Group）を構成し、各ECCグループは例えば3つのデータディスクと1つのパリティディスクで構成される。更に、後述のように、このようなアレイドディスク5の1組に対し、二重化されたホストアダプタ二重化されたホストアダプタと二重化されたディスクアダプタを通じて、あるCPUからアクセスできるようになっている。そして、ホストアダプタの一方に障害が発生したときには、ホストアダプタの他方もしくはディスクアダプタの他方を通じて、同じCPUから同じアレイドディスクにアクセスすることができる。

【0039】キャッシュメモリパッケージ3において、18は各アダプタのMP10が共通にアクセス可能で種々の管理情報を記憶する共有メモリ部、19は共有メモリ制御部、20はキャッシュメモリ部、21はキャッシュメモリ制御部であり、両メモリ制御部19、21は共にメモリ書き込みデータ保証の為にECC生成回路、読み出しデータの検査及び訂正回路を内蔵し、キャッシュメモリパッケージ3全体で最大1GBのキャッシュ容量を実現しており、装置構成上は2面化して実装している。

【0040】キャッシュメモリ容量を更に増設する場合は、キャッシュメモリパッケージ3の代わりに（または、キャッシュメモリパッケージ3に加えて）22で示すキャッシュポートパッケージを実装し、23で示すプラッタ（基板差し込み板）間接続ケーブルを介して24で示すキャッシュユニットに接続し、（すなわち、増設

ユニット24内のキャッシュメモリには、キャッシュポートパッケージ22及びケーブル23を介してアクセスできるように構成され）、これによって、最大8GB2面までキャッシュ容量を増設することができる。図2では、キャッシュメモリパッケージ2を2面設けたのに加えて、キャッシュポートパッケージ22を実装し、これにケーブル24を介していくつかのキャッシュユニット24を接続した場合を示している。

【0041】以上述べたホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3はコモンバス4を介してつながっているが、このコモンバス中、25は各アダプタのMP10が共有メモリをアクセスするためのマルチプロセッサバス（以下「Mバス」という）、26は高速データ転送を行う高速I/Oバス（以下「Fバス」という）である。

【0042】高速I/Oバス26は通常は64ビット幅で2系統同時に動作しているが、障害発生時はどちらか1系統のみでの縮退動作が可能であり、またMバス25に障害が発生した場合はFバス26のどちらか1系統を使用して動作可能である。

【0043】更に活線挿抜対応（挿抜の際、挿抜部品の負荷を小さくして挿抜を行なうことで、システムを稼働状態のまま挿抜を可能とする）のBTL9をコモンバス4のインターフェイスにすることで、ホストアダプタ1に障害が発生した場合、システムは自動的に本障害バスを閉塞し他のホストアダプタのバスを用いてアレイドディスク5に対し対上位（同じCPU）からのアクセスを継続する。保守員は、システム稼働状態において障害の発生したホストアダプタ1を取り除き、正常なホストアダプタ1をシステムに挿入し、27の保守用プロセッサ（以下「SVP」という）から28のLANを介して復旧の指示を与え、システムは交換されたホストアダプタ1の動作をチェックし正常であれば閉塞バスを復旧させることにより、無停止運転を実現している。なお、図中LANCは、LAN Controller（SVPインタフェースコントローラ）である。SVP27は、他のホストアダプタ及びディスクアダプタにも同様に接続され、監視及び保守が行なわれるようになっている。

【0044】また、各アダプタの制御プログラムに変更がある場合は、SVP27からLAN28を介してブートデバイス12内にある制御プログラムの内容を書き替えることにより無停止のアップグレードが可能である。

【0045】即ち、システムの制御プログラムをアップグレードを実施する場合は、まずホストアダプタ/ディスクアダプタの各モジュールを1モジュールずつ閉塞し、制御プログラムのアップグレードを行い再接続する。以上のように1モジュールずつの制御プログラムの入れ換え操作を繰り返すことにより、系全体の制御プログラム入れ換えが実施される。

【0046】図3は、図2に示した構成図に沿ってデー



タの流れとデータの保証を示した図である。

【0047】上位からアレイディスクにデータを書き込む場合、例えばESCON（光チャンネルの商標名、IBM社）から、先ず書き込み先の記憶空間上の物理アドレス情報（以下「PA」という）が送られて来た後、データ（CKD（Count Key Data）フォーマット）+CRCコードが送られてくる。これらの光信号は信号変換部6で電気信号に変換すると共にパリティを生成し、フォーマット変換部7ではデータフォーマットをFBA（Fired Blocked Architecture）フォーマットに変換すると共にLRC（Longitudinal Redundancy Check、長手方向冗長度チェック）コードを付加し、更にPAをデータの一部として取り込んでアレイディスク上の論理アドレス（以下「LA」という）を生成した後これら総ての情報に対してパリティを付加してFバス26に送られる。

【0048】キャッシュパッケージ3では、Fバス26からのデータに対して誤り訂正可能なECCを付加してキャッシュメモリ20に書き込む。

【0049】ディスクアダプタ2では、Fバスからのデータに対して更にCRCコードが付加され、該データSCIインターフェースを介してアレイディスク5に送られ、磁気ディスク装置個々にECCを付加して書き込みデータを保証している。

【0050】アレイディスク5からのデータ読み出しにおいても同様に、各チェックコードを元に読み出しデータの検査/訂正を行い信頼性を高めている。

【0051】以上のように、チェックコードはデータの長さ方向に対してはある長さ毎の水平チェック、データの垂直（幅）方向に対しては（例えばバイト単位の）垂直チェックで2重化されており、また転送が行われる領域間（図中一点鎖線）では当該2重化チェックコードのうち1つを必ずデータとして受け渡すことによりデータ保証に万全を期している。

【0052】図4は図1で述べたスケーラビリティを実現するための装置外観図であり、41はアレイディスクを制御する制御ユニット部、42はアレイディスクを実装するアレイユニット部で、本装置はこの2つのユニットで構成される。

【0053】図5は制御ユニット41の実装図で（a）は正面図、（b）は側面図を表わす。51はホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3を実装する論理架部、52は停電時に揮発メモリであるキャッシュメモリ部に電源を供給するバッテリー部、53はキャッシュメモリ増設時にキャッシュユニット24及び増設メモリ用の追加バッテリーを実装するキャッシュメモリ増設部、54はSVP実装部、55は論理架に電源を供給する論理架用スイッチング電源、56はアレイディスクの構成（容量）が小規模の場合のアレイ

ディスク実装部、57はアレイディスク部に電源を供給するアレイディスク用スイッチング電源、58は両スイッチング電源55、57に電源を供給する商用電源制御部である。

【0054】図6は大容量アレイディスクを構成するときのアレイユニット部の実装図で（a）は正面図、（b）は側面図を表わす。

【0055】アレイディスク実装部56は、磁気ディスク装置を最大112台（8行x7列x2）実装可能であり、各磁気ディスク装置に障害が発生した場合の装置の入れ替えを容易にするために、装置の正面と背面の両面から挿抜可能となるような実装方式をとっている。

【0056】61はユニット全体の発熱を逃がすための冷却ファンで、冷却効果を高めると共に、騒音抑止の観点から小さな冷却ファンを使って小区分化し、床面より天井へ送風する構造をとっている。

【0057】図7は図5で説明した論理架部の接続方式図である。

【0058】71はコモンバス4をプリント配線したブラッタ（基板の挿し込み用の板）であり、72は各アダプタ、パッケージとブラッタ71を接続するためのコネクタである。

【0059】ホストアダプタ1、ディスクアダプタ2、キャッシュメモリパッケージ3の間のデータ転送はコモンバス4を介して行うため、各アダプタ、パッケージはコネクタ72上の任意のどの位置でも接続可能となり、ホストアダプタ1の実装数、ディスクアダプタ2の実装数を自由に変えることができる。

【0060】一方、キャッシュ容量を増設する場合はキャッシュメモリパッケージ3をキャッシュポートパッケージ22に変えて実装するか、または図7に示すように、キャッシュメモリパッケージ3に加えてキャッシュポートパッケージ21を実装し、これに、接続ケーブル23を介してキャッシュユニット43（図2の24に相当）に接続することにより、もとの2GBの容量に加えて更に最大8GB2面分のキャッシュメモリ容量を拡張できる。

【0061】図8は図5で示した論理架部の実装イメージ図である。

【0062】図8で、コモンバス4は、ブラッタ71上を左右方向にプリント配線されており、このブラッタ71に対して、キャッシュポートパッケージ22の基板（CP）の取付部、キャッシュメモリパッケージ3の基板（C）の取付部、ホストアダプタモジュールの基板（H）の取付部、及びディスクアダプタモジュールの基板（D）の取付部が設けられ、図の矢印84で示すように、各基板は、挿抜操作面側から着脱されるようになっていて、ブラッタ71に差し込まれるとコモンバス4と電気接続されるものである。

【0063】81は、ホストアダプタ1の基板上の下方

部に実装されて、対上位インターフェイスを司る光コネクタ部、82はディスクアダプタ2の基板上の下方部に実装されて、アレイディスク5と接続するSCSIコネクタ部、83はキャッシュポートパッケージ22を実装したときの接続ケーブル23用の接続コネクタ部である。85は、キャッシュメモリパッケージ3の基板(C)の下方部に取付けたキャッシュメモリ本体(図2のキャッシュメモリ20)である。

【0064】各コネクタ部は、障害発生等で各アダプタ、パッケージを挿抜する際の操作性を向上させるため、接続コネクタ83を除き、操作面84側へは実装せず、プラッタ71の接続側に集中実装している。

【0065】図9は本発明のソフトウェア構成を示した図である。

【0066】91はホストアダプタ1のブートデバイス12に書き込まれるチャンネルアダプタ制御プログラム(以下「CHP」という)、である。また、ディスクアダプタ2のブートデバイス12に書き込まれるディスクアダプタ制御プログラムのうち、92はアレイディスク固有の処理およびキャッシュメモリとアレイディスク間のデータ転送制御を受け持つディスクアダプタマスタ制御プログラム(以下「DMP」という)、93はDMP92の制御管理下でキャッシュメモリ20とアレイディスク5の間のデータ転送制御を受け持つディスクアダプタスレーブ制御プログラム(以下「DSP」という)である。

【0067】ディスクアダプタ2のブートデバイス12には、DMP92とDSP93の2種類が書き込まれているが、装置構成上nセットのディスクアダプタでアレイディスクにアクセスする場合、そのうちの2セットがDMP92として動作(2重化)し、残るn-2のディスクアダプタがDSP93として動作する。

【0068】94はSVP27に搭載するSVP制御プログラムで、CHP91、DMP92、DSP93を監視及び保守するとともに、各制御プログラムの更新時はSVP27から更新したいMPの制御プログラムを直接、または他のMPから当該MPの制御プログラムを更新することができる。

【0069】図10はデータの流りに基づいた図9で示したソフトウェア構成の機能分担を示した図である。

【0070】CHP91は、上位からのアドレス形式及びデータ形式を下位アドレス形式及びデータ形式に変換し、キャッシュメモリに書き込む。101はセグメント、102はブロック、103はアレイディスク5上の磁気ディスク1台当りに書き込むデータ量を表すストライプである。DMP92は、キャッシュメモリ上からストライプ単位にデータを読み出し、下位アドレスをアレイディスクの行NO、列NO、FBA、ブロック数に変換し、DSP93でアレイディスクにデータを書き込む。

【0071】また、DMP92はアレイディスク5の構成情報も管理している。

【0072】以上のように、各制御プログラムを機能分担することにより、上位インタフェースをSCSIやファイバーチャネル等に変更する場合はCHP91のみ、またアレイディスク構成を変更(ディスクの行数/列数、RAID(Redundant Array Inexpensive Disk)方式等)する場合はDMP92のみの変更で対応可能であり、ホストアダプタ1、ディスクアダプタ2の接続変更に合わせて各制御プログラムを書き替えることで、スケラビリティを実現するとともに、ソフトウェア開発の負荷も軽減している。

【0073】図11はコモンバス4の2重化の考え方と縮退動作を説明した図である。

【0074】111はコモンバス4の使用権を獲得することのできるバスマスタ(MP10を搭載しているホストアダプタ1又はディスクアダプタ2)、112はバスマスタ111からのアクセス要求を受けるバススレーブ(キャッシュメモリパッケージ)である。

【0075】Fバス26は通常動作状態では64ビットバス(200MB/S)2系統を同時に動作させ400MB/Sを実現しており、各バス系統はパリティチェック又はタイムアウトで障害を検出可能である。障害発生時はバスマスタ111は各自縮退状態に入り、残る1系統を使ってバススレーブをアクセスすると共に、この時の縮退情報は共有メモリ18上の管理エリアに登録される。

【0076】またコモンバス内のシステム制御信号(バスリセット等)は信号線を3重化しており、通常動作時は3線一致、縮退動作時は2線一致(多数決)方式を採用することにより信頼性を高めている。

【0077】図12は装置各部位における多重化と縮退運転を示した図である。

【0078】121は2ポート化されたチャンネルバスであり、ホストアダプタ1にはチャンネルアダプタ29が2モジュール、対上位用のチャンネルバスが4バス実装しており、障害発生時は交替チャンネルアダプタ(CHP)、交替チャンネルバスを使用して縮退運転に入る。

【0079】122はディスクアダプタ2とアレイディスク5の間のインタフェースを司るSCSIバスで、1行の磁気ディスク群に対して別のディスクアダプタ2からもアクセス可能なように2重化しており、当バスに障害が発生した場合は交替SCSIバスを使用して縮退運転に入る。また、アレイディスクマスタ制御を行うDMP92も2重化しており、障害発生時は交替DMP92を使用して縮退運転に入る。

【0080】共有メモリ18、キャッシュメモリ20も2重化しており、共有メモリに障害が発生した場合は残るもう一方の使用して縮退運転に入り、キャッシュメモ

りに障害が発生した場合はライトペンディングデータ（キャッシュメモリ上に残っているデータ）をディスクにデステージし障害発生メモリ部位を除いたメモリで縮退運転を行う。

【0081】アレイディスク5上の磁気ディスクに障害が発生した場合は、当該磁気ディスクを切り離し予備の磁気ディスクに修復しながら読み出し書き込み動作を行う。

【0082】図13は装置の電源系の多重化と縮退運転を示した図である。

【0083】商用電源制御部58は各々独立したAC入力で2重化して、論理架用スイッチング電源55及びアレイディスク用スイッチング電源57にそれぞれ供給しているため、障害発生時はもう片方の商用電源制御部58で縮退運転に入る。

【0084】131は上位ホストからの電源ON/OFFの遠隔制御や商用電源制御部58、両スイッチング電源等の電源回路を制御する電源制御回路（以下「PCI」という）である。

【0085】論理架用スイッチング電源55は冗長運用として必要数より2回路多く実装し電源コモンバスを介して論理架51及びバッテリー52に供給することにより、当スイッチング電源55が2回路故障しても動作可能である。

【0086】同様に列単位の磁気ディスク群に供給するにアレイディスク用スイッチング電源57も、冗長運用として2回路多く実装し電源コモンバスを介して供給することにより、当スイッチング電源57が2回路故障しても動作可能であり、さらに両スイッチング電源55、57を2重化するよりも安価な構成に仕上げることができる。

【0087】また停電時においては、2重化されたバッテリー52から電源コモンバスを介して論理架内の揮発メモリであるキャッシュメモリ及びPCI131に供給され、片方のバッテリーが故障しても動作可能である。

【0088】図14及び図15はアレイディスクに使用する磁気ディスク装置単体の記憶容量別にアレイディスクを構成したときのシステム性能を比較した図である。

【0089】図14はそれぞれ異なる磁気ディスク装置を使用して同一容量のアレイディスクを実現した場合の構成を示しており、項番141が3GBの磁気ディスク装置（3.5インチ径のディスクを使用）、項番142が4.0GBの磁気ディスク装置（5インチ径のディスクを使用）、項番143が8.4GBの磁気ディスク装置（6.4インチ径のディスクを使用）を使用している。アレイ構成は、ディスク装置141が14枚のデータディスクの2枚のパーティティディスク、ディスク装置142が14枚のデータディスクと4枚のパーティティディスク、ディスク装置143が14枚のデータディスクと2枚のパーティティディスクで構成した場合である。

【0090】図15は各磁気ディスク装置141、142、143についての毎秒当りのI/O命令発行件数と平均応答時間の関係を示しており、アレイディスクシステムとしてのトランザクション性能を向上させるためには、小容量（小径）の磁気ディスク装置を使用してアレイ構成を大きくすることが最も性能を引き出せることから、本発明に於ては3.5インチ磁気ディスク装置141を採用してアレイディスクシステムを実現している。従って、同じ記憶容量の磁気ディスク装置を、従来のように大形磁気ディスク装置1台で構成するのと、複数台の小形磁気ディスク装置のアレイで構成するのでは、後者の小形磁気ディスク装置を多数用いたアレイ構成のものの方が、平均アクセスタイムを短縮できる点で有利である。

【0091】以上説明してきたスケラブルなアーキテクチャを使用して実現できる装置モデル構成例を図16～図19に示す。

【0092】図16は、コモンバス4上のディスクアダプタ2の実装数を減らし、更にキャッシュポートパッケージ22を実装し、接続ケーブル23を介してキャッシュユニット24に接続することにより、キャッシュヒット率の高める高性能大容量キャッシュメモリ付小形ディスクアレイを実現した時の構成図である。

【0093】またディスクアダプタ2を実装しないで、ホストアダプタ1とキャッシュメモリのみで構成した場合（図中の破線内の構成）は、記憶媒体が磁気ディスクから半導体メモリに代わり、更に高速データ転送可能な高性能の半導体ディスク装置を実現する。

【0094】図17はディスクアダプタ2を最大構成とし、キャッシュパッケージ3を実装し又はキャッシュポート22を実装し接続ケーブル23を介してキャッシュユニットを接続することにより、高性能大容量キャッシュメモリ付大形ディスクアレイを実現した時の構成図である。

【0095】図18はホストアダプタ1の対上位インターフェースをSCSI/ファイバチャネル等のインターフェースに変えて、ディスクアダプタ2の実装数を減らし、更にFバス26のビット幅を半分にした2系統で構成することにより、オープン市場をターゲットにした無停止運転の高性能フォールトトレラント（高信頼性）サーバシステムを実現した時の構成図である。

【0096】図19は図18の構成を元に2重化、活線挿抜を考慮せずに、最もシンプルな構成をとることによって安価なオープン市場向けのサーバシステムを実現した時の構成図である。なお、図中、4D+1Pは、データディスク4枚とパーティティディスク1枚の趣旨である。

【0097】以上の実施例において、コモンバス4上に、更に光ディスクアダプタ（光ディスク用接続論理モジュール）を介して光ディスク装置を接続し、磁気テープ制御装置（磁気ディスク接続論理モジュール）を介し

て磁気テープ装置を接続し、あるいは半導体記憶装置接続論理モジュールを介して半導体記憶装置を接続することができる。また、コモンバス4上に別の形式のホストアダプタを介してワークステーションを接続することもできる。このように、コモンバス上に、種々の形式の記憶装置に対する記憶媒体アダプタを接続することができる。

【0098】

【発明の効果】以上詳しく説明したように、本発明によれば、上位装置に対するインタフェースを構成する複数の上位側接続論理装置と、記憶装置と、前記記憶装置に対するインタフェースを構成する複数の記憶装置側接続論理装置と、これらの装置間で転送されるデータを一時記憶するキャッシュメモリ装置（複数の上位側接続論理装置及び複数の記憶装置側接続論理装置に共有されるキャッシュメモリ装置）とを有する記憶システムにおいて、前記複数の上位装置側接続論理装置、複数の記憶装置側接続論理装置、及びキャッシュメモリ装置は、これらの装置に共有されるコモンバスにより相互に接続されるように構成したので、上位側接続論理装置と記憶装置側接続論理装置とキャッシュメモリの増設または変更は、単にコモンバス上にこれらの装置等を追加または変更して行くだけでよく、増設によるアップグレードが容易に達成できスケラブルなシステム構成を得ることができる。また、これらの上位側接続論理装置、記憶装置側接続論理装置及びキャッシュメモリ装置は、モジュール化されて、コモンバスの配設されたプラッタに挿抜（着脱）自在に取り付けるようにしたので、これらの装置の必要な数量の増設作業も簡単であるという効果がある。

【0099】また、上位側接続論理装置、記憶装置側接続論理装置、キャッシュメモリ装置、及びこれらの間を接続するコモンバスは、二重化され、2系統に分けて配線されているので、これらの装置の一方に障害が発生したときでも、他方の装置を用いて縮退運転が可能である。この場合、上位側接続論理装置、記憶装置側接続論理装置、及びキャッシュメモリ装置は、いずれも活線挿抜対応のコネクタ部を具備しているため、システムを停止することなく保守点検を行なって故障部品の交換を行ったり、増設用の部品を追加したりすることが可能であるという効果がある。

【0100】更に、記憶装置は、複数の小形記憶装置を組み合わせたアレイ形とされ、これにより従来の大形ディスク装置1台を用いたものに比べてアクセスタイムを短縮できるという効果がある。

【0101】また、キャッシュメモリ装置は、コモンバ

スに直接取り付けられるキャッシュメモリモジュール（キャッシュメモリパッケージ）と、増設用のキャッシュユニットとで構成され、増設用のキャッシュユニットは、コモンバスに直接挿抜自在に取り付けられる増設用のキャッシュポートパッケージを介して必要数接続されるようになっているので、簡単に増減することができるという効果も得られる。

【0102】以上により、高信頼性の記憶システムを得ることができる。

10 【図面の簡単な説明】

【図1】本発明の実施例の概要を示す概念図である。

【図2】本発明の一実施例の記憶システムの詳細な構成図である。

【図3】図2の構成図に沿ったデータの流れとデータ形式を示した図である。

【図4】本発明の一実施例の装置外観図である。

【図5】本発明の一実施例の装置における制御ユニット部の実装方式図である。

【図6】本発明の一実施例の装置におけるアレイディスクユニット部の実装方式図である。

20

【図7】本発明の一実施例の装置における論理架部の接続方式図である。

【図8】本発明の一実施例の装置における論理架部の実装方式図である。

【図9】本発明の実施例に適用されるソフトウェア構成図である。

【図10】本発明の実施例によるデータの流れとソフトウェアの機能分担を示した図である。

30

【図11】本発明の実施例によるコモンバスの2重化と縮退動作を示した図である。

【図12】本発明の実施例による装置各部位の2重化と縮退運転を示した図である。

【図13】本発明の実施例による装置の電源系の多重化と縮退運転を示した図である。

【図14】アレイディスクに使用する磁気ディスク装置単体のディスク構成を示す図である。

【図15】磁気ディスク装置の記憶容量とアレイディスクのシステム性能を示した図である。

40

【図16】高性能大容量キャッシュメモリ付小形ディスクアレイの構成図である。

【図17】高性能大容量キャッシュメモリ付大形ディスクアレイの構成図である。

【図18】高性能フォールトトレラントサーバシステムの構成図である。

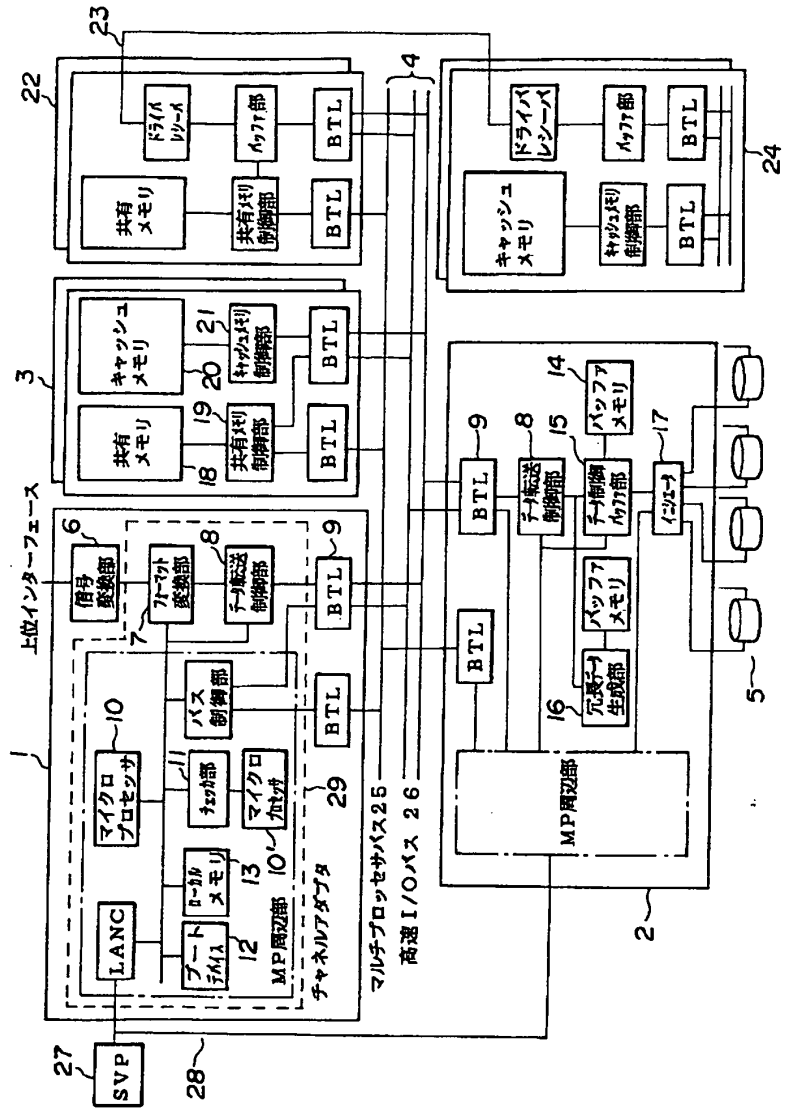
【図19】低価格サーバシステムの構成図である。

【図20】従来の記憶システムの概略構成図である。



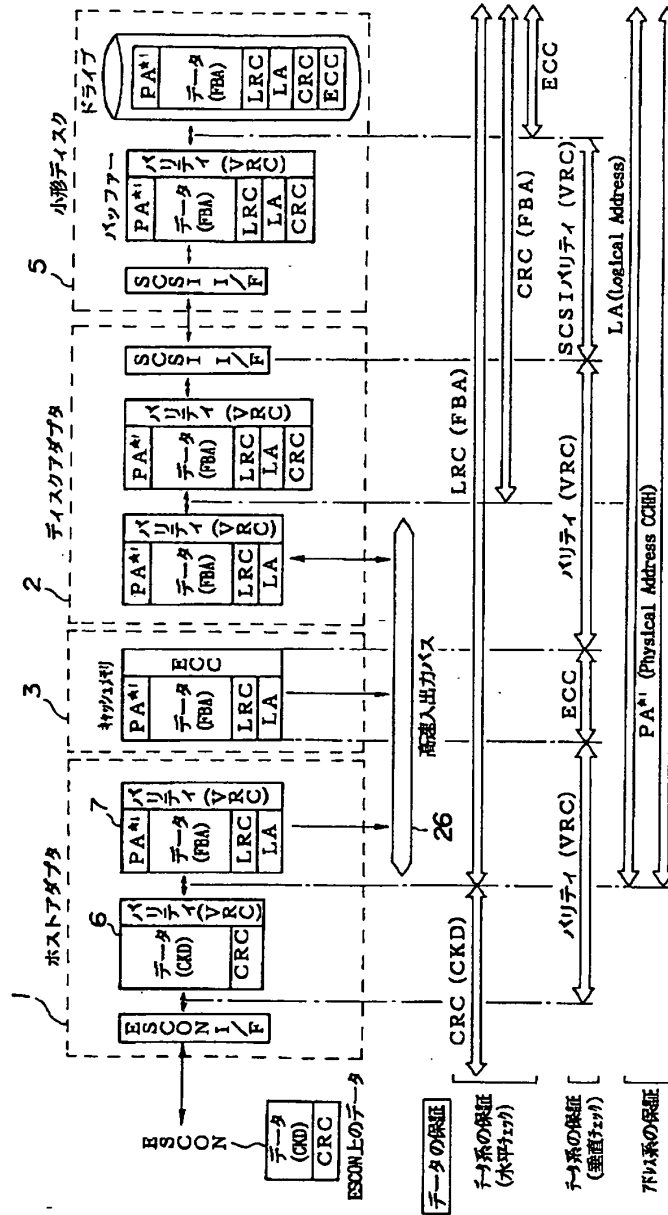
【図2】

【図2】



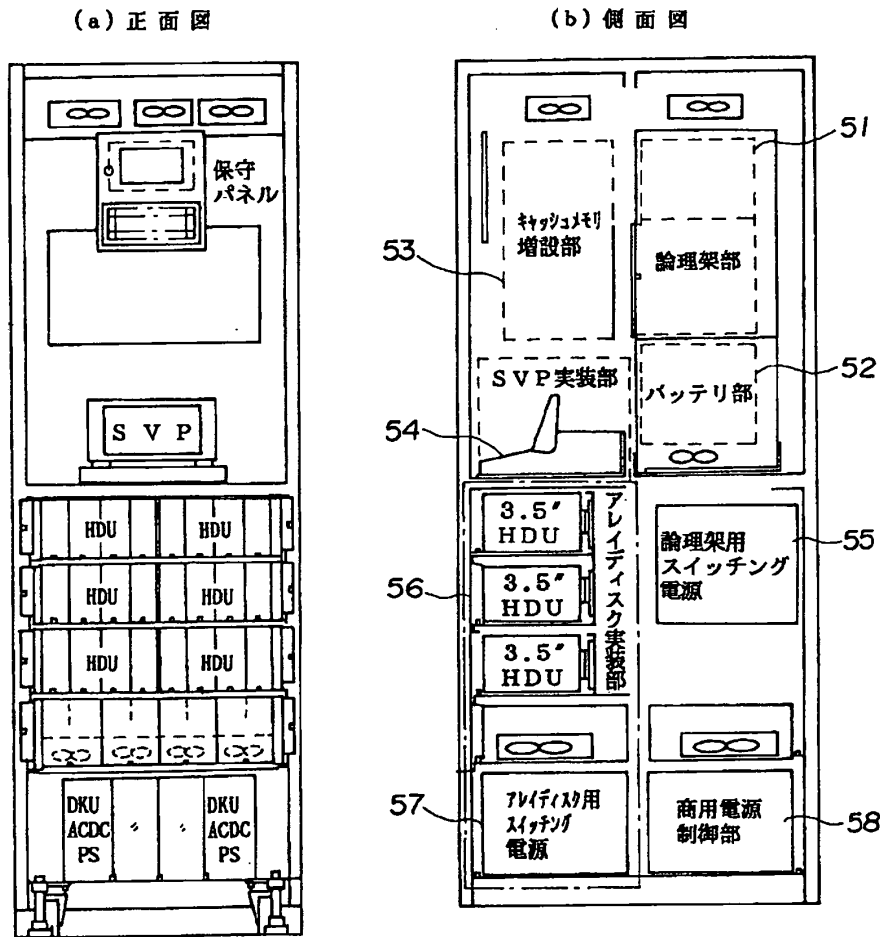
【図3】

【図3】



【図5】

【図5】



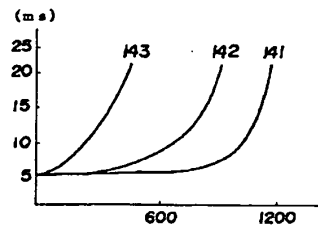
【図14】

【図15】

【図14】

【図15】

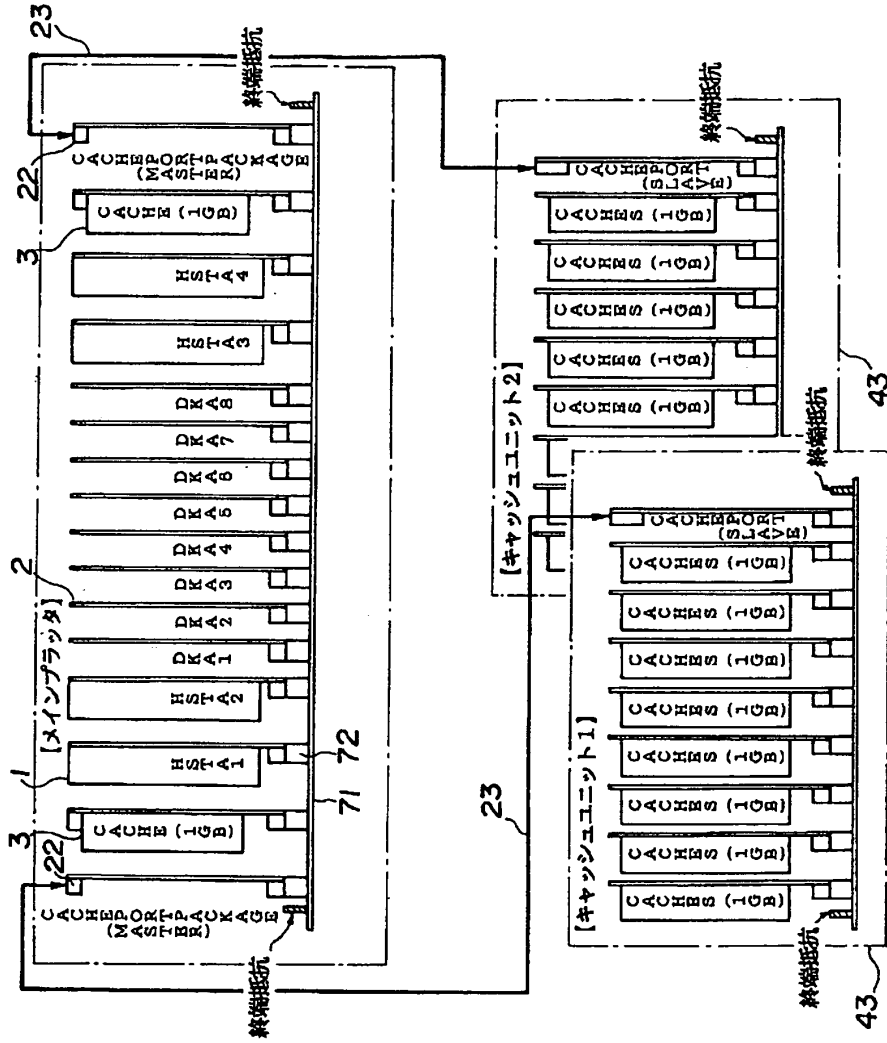
項番	磁気ディスク単体容量	アレイ構成	アレイ容量
141	3.0GB (3.5インチ)	(14D+2) × 6	約220GB
142	4.0GB (5インチ)	(14D+2) × 4	
143	8.4GB (6.4インチ)	(14D+2) × 2	





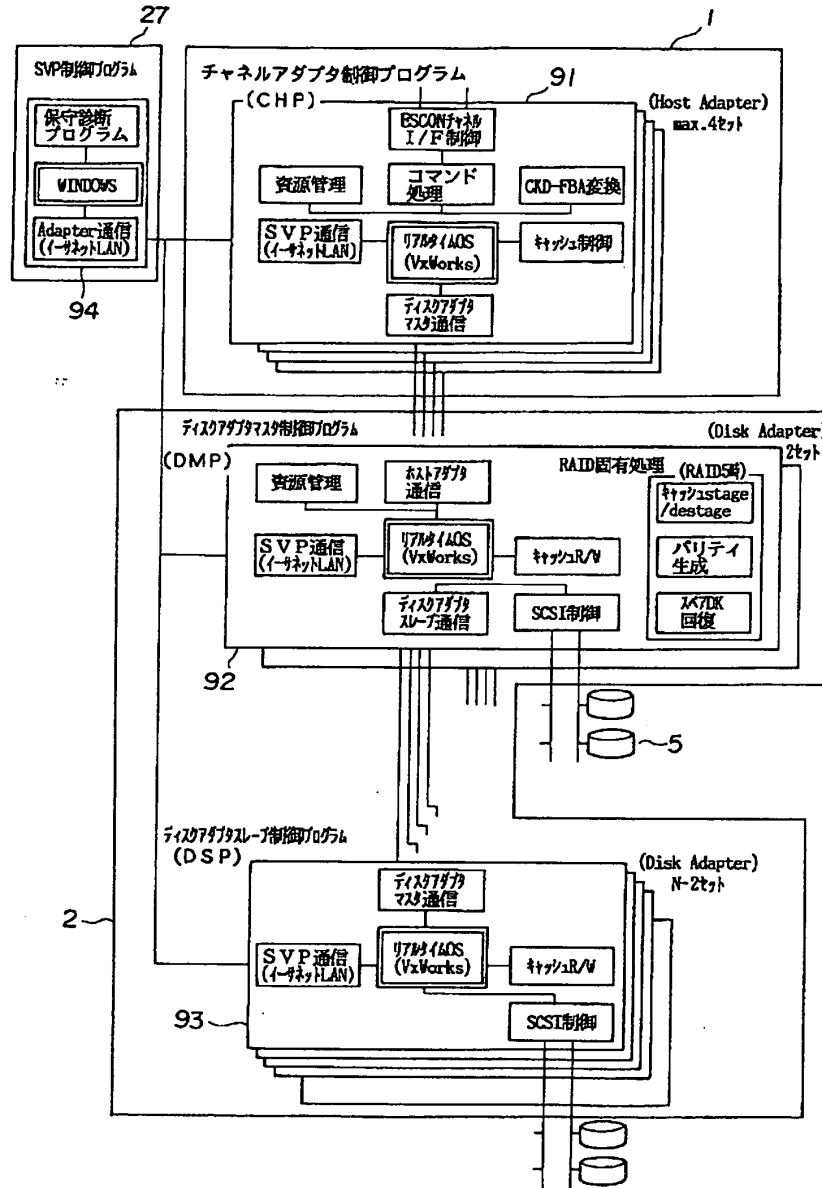
【図7】

【図7】

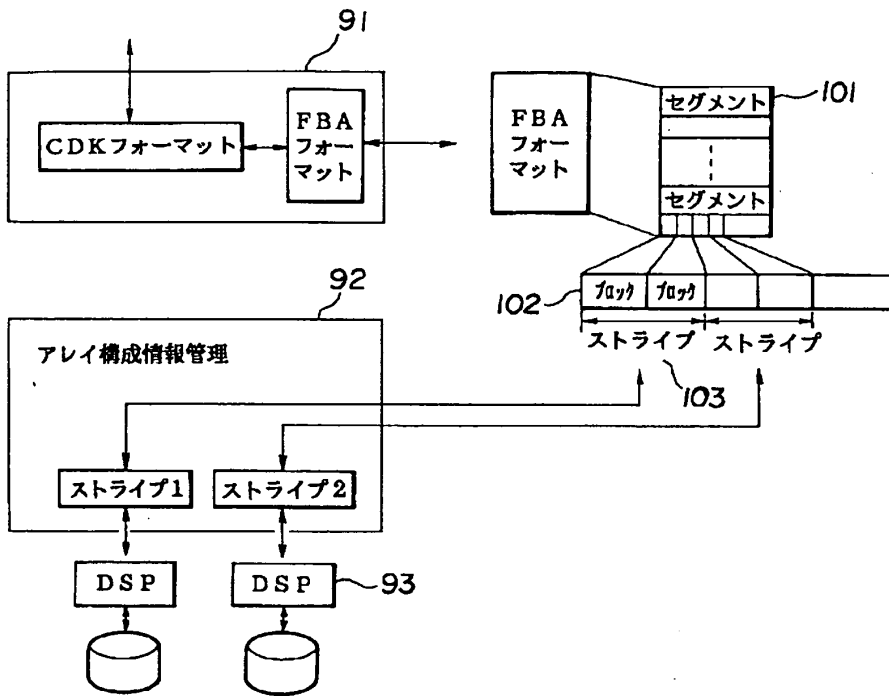


【図9】

【図9】



【図10】



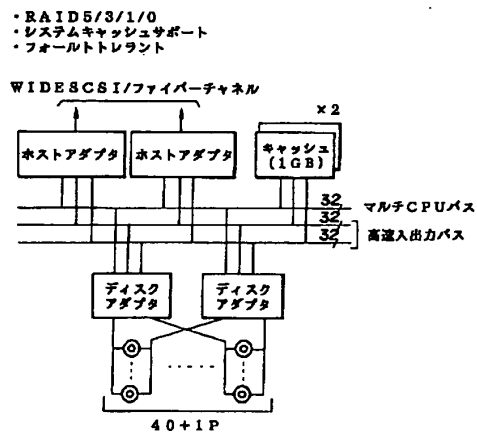
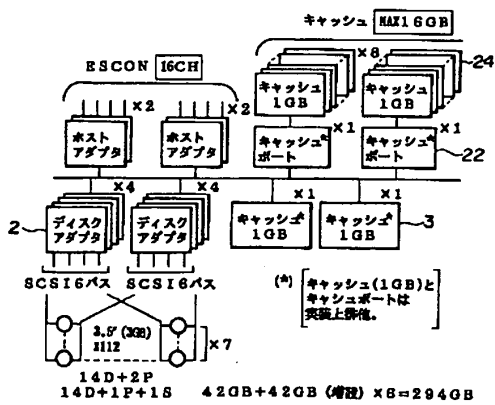
【図10】

【図17】

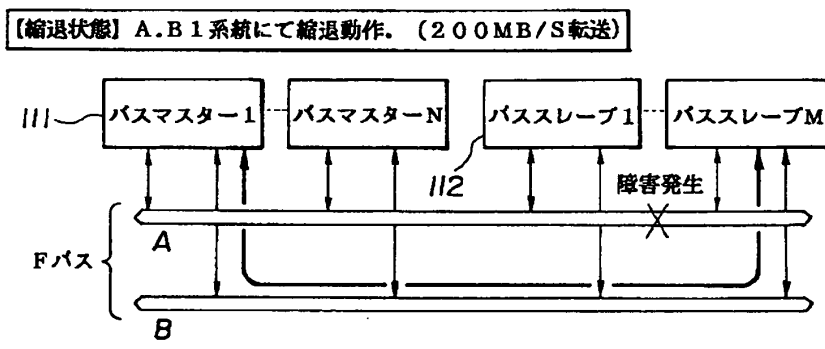
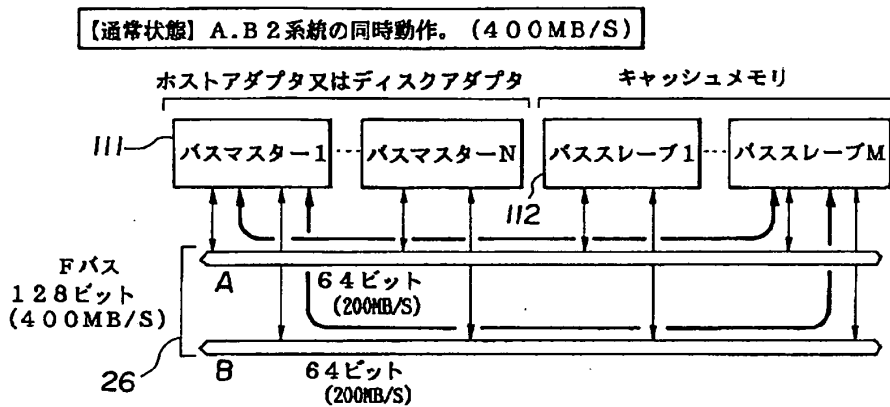
【図18】

【図17】

【図18】

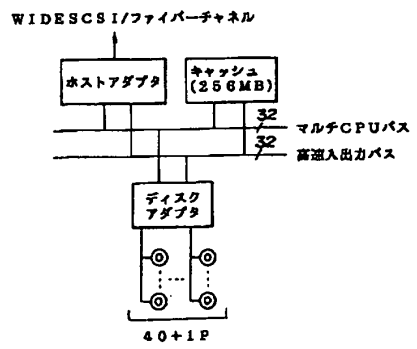


【図11】



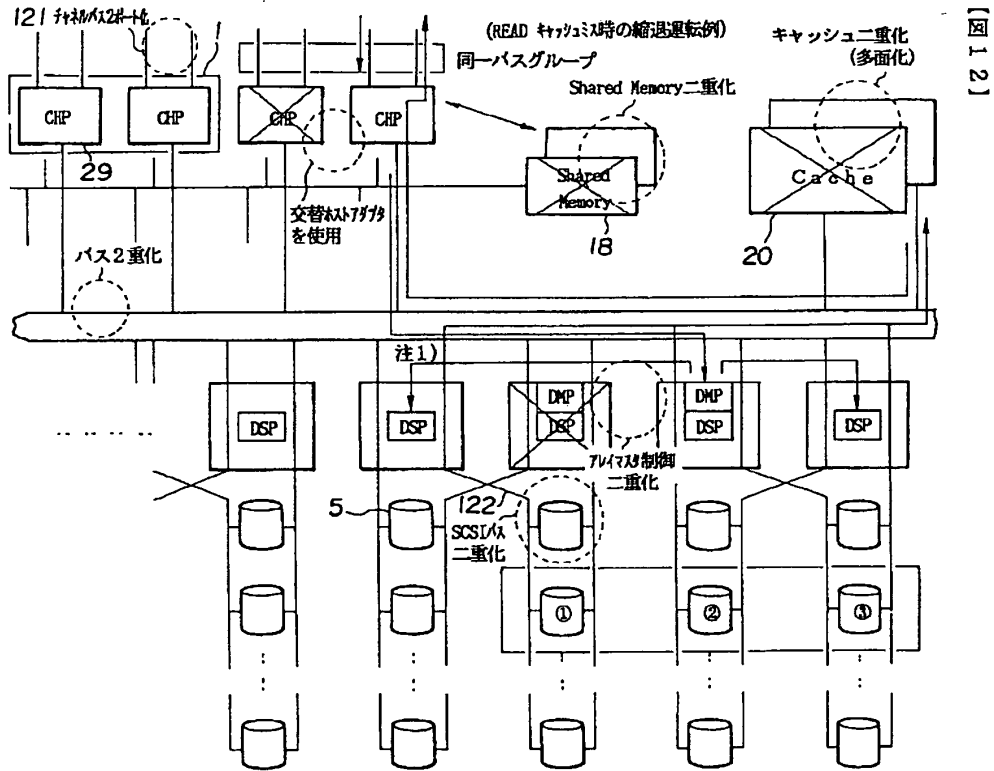
【図19】

【図19】



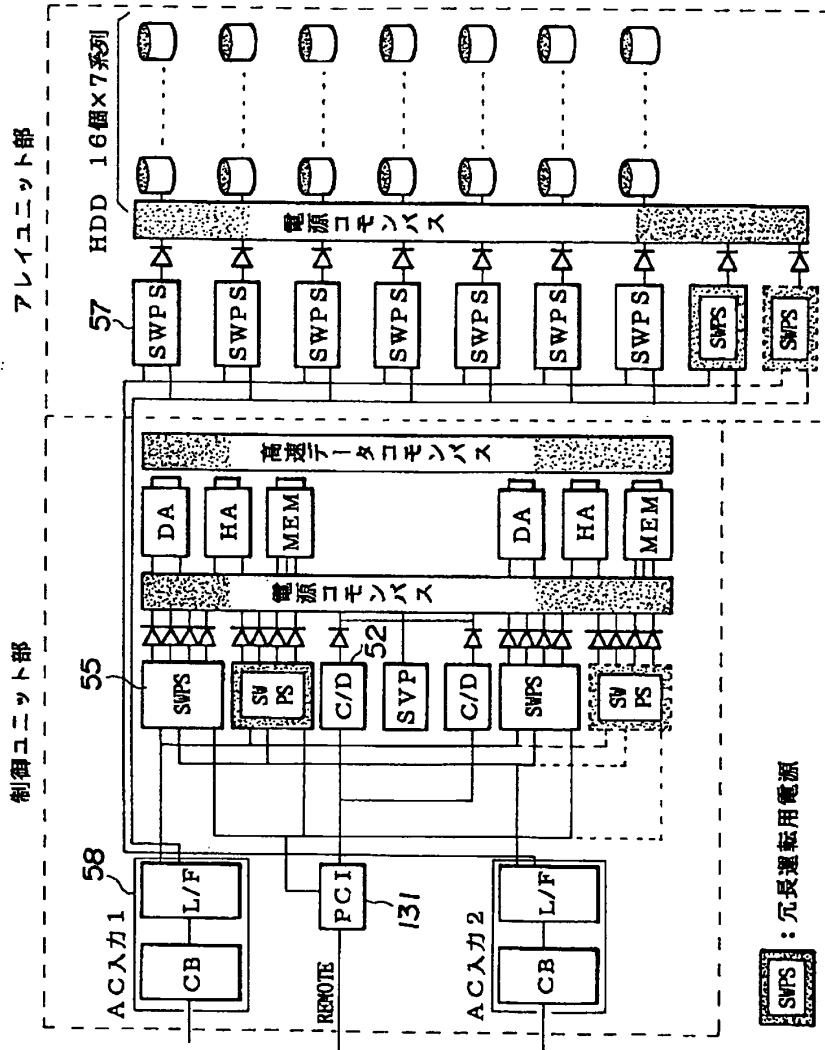
【図11】

【図12】



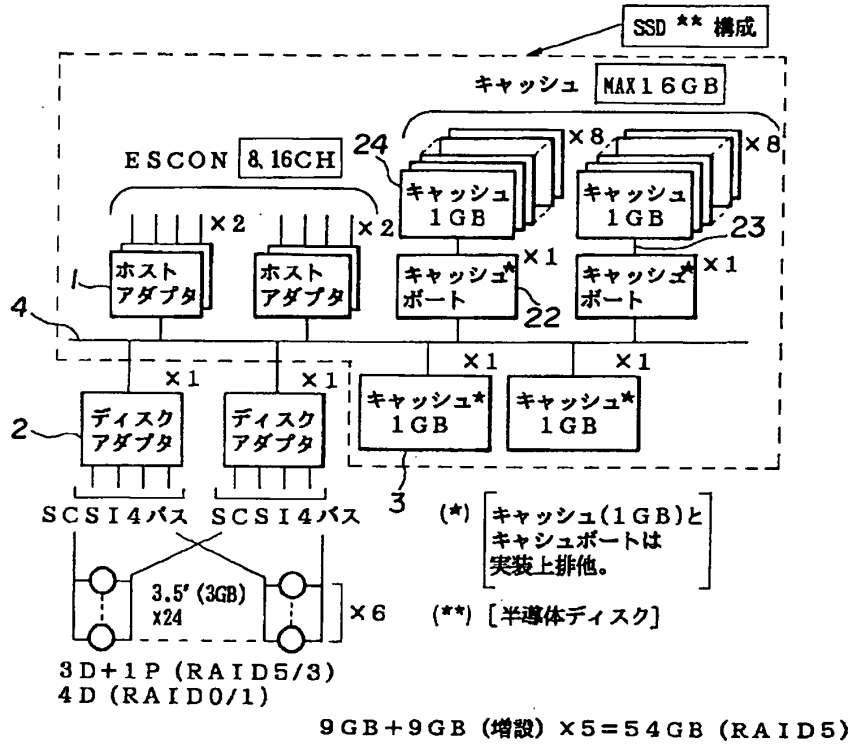
【図13】

【図13】

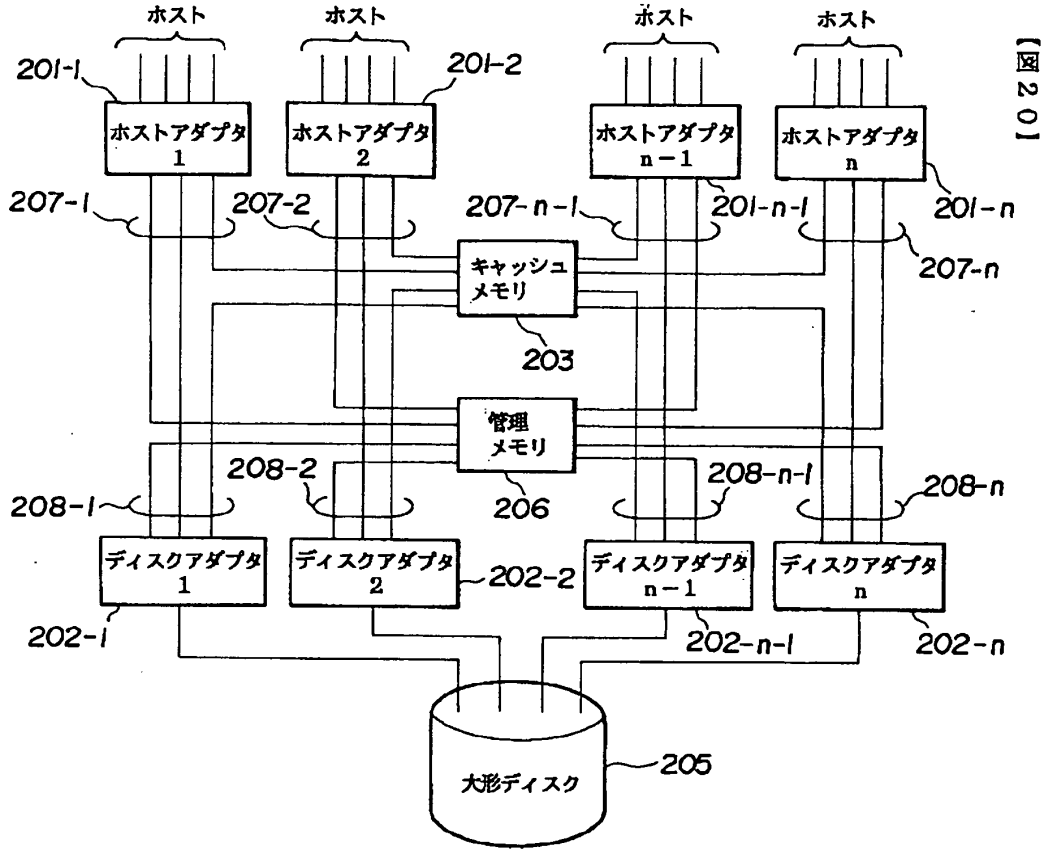


【図16】

【図16】



【図20】



フロントページの続き

(72)発明者 高橋 直也  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 井上 靖雄  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 岩崎 秀彦  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 星野 政行  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内

(72)発明者 磯野 聡一  
 神奈川県小田原市国府津2880番地 株式会  
 社日立製作所ストレージシステム事業部内



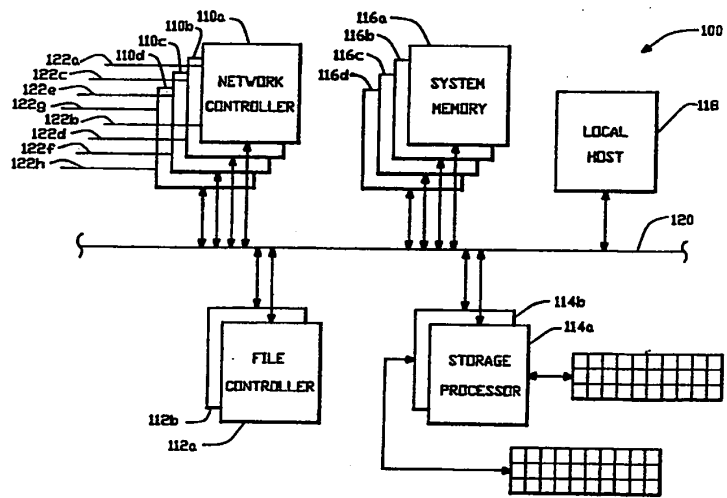


B10

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification <sup>5</sup> : <b>G06F 15/16</b></p>	<p><b>A1</b></p>	<p>(11) International Publication Number: <b>WO 91/03788</b> (43) International Publication Date: 21 March 1991 (21.03.91)</p>
<p>(21) International Application Number: PCT/US90/04711 (22) International Filing Date: 20 August 1990 (20.08.90) (30) Priority data: 404,959 8 September 1989 (08.09.89) US (71) Applicant: AUSPEX SYSTEMS, INC. [US/US]; 2952 Bunker Hill Lane, Santa Clara, CA 95054 (US). (72) Inventors: ROW, Edward, John ; 468 Mountain Laurel Court, Mountain View, CA 94064 (US). BOUCHER, Laurence, B. ; 20605 Montalvo Heights Drive, Saratoga, CA 95070 (US). PITTS, William, M. ; 780 Mora Drive, Los Altos, CA 94022 (US). BLIGHTMAN, Stephen, E. ; 775 Salt Lake Drive, San Jose, CA 95133 (US).</p>	<p>(74) Agents: FLIESLER, Martin, C. et al.; Fliesler, Dubb, Meyer &amp; Lovejoy, 4 Embarcadero Center, Suite 400, San Francisco, CA 94111 (US). (81) Designated States: AT (European patent), AU, BE (European patent), CA, CH (European patent), DE (European patent)*, DK (European patent), ES (European patent), FR (European patent), GB (European patent), IT (European patent), JP, KR, LU (European patent), NL (European patent), SE (European patent). <b>Published</b> <i>With international search report. Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.</i></p>	

(54) Title: PARALLEL I/O NETWORK FILE SERVER ARCHITECTURE



(57) Abstract  
A file server architecture is disclosed, comprising as separate processors, a network controller unit (110), a file controller unit (112) and a storage processor unit (114). These units incorporate their own processors, and operate in parallel with a local Unix host processor (118). All networks are connected to the network controller unit (110), which performs all protocol processing up through the NFS layer. The virtual file system is implemented in the file controller unit (112) and the storage processor (114) provides high-speed multiplexed access to an array of mass storage devices. The file controller unit (112) controls file information caching through its own local cache buffer, and controls disk data caching through a large system memory which is accessible on a bus by any of the processors.

\* See back of page

### DESIGNATIONS OF "DE"

Until further notice, any designation of "DE" in any international application whose international filing date is prior to October 3, 1990, shall have effect in the territory of the Federal Republic of Germany with the exception of the territory of the former German Democratic Republic.

#### *FOR THE PURPOSES OF INFORMATION ONLY*

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	ES	Spain	MC	Monaco
AU	Australia	FI	Finland	MG	Madagascar
BB	Barbados	FR	France	ML	Mali
BE	Belgium	GA	Gabon	MR	Mauritania
BF	Burkina Fasso	GB	United Kingdom	MW	Malawi
BG	Bulgaria	GR	Greece	NL	Netherlands
BJ	Benin	HU	Hungary	NO	Norway
BR	Brazil	IT	Italy	PL	Poland
CA	Canada	JP	Japan	RO	Romania
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo			SE	Sweden
CH	Switzerland	KR	Republic of Korea	SN	Senegal
CM	Cameroon	LJ	Liechtenstein	SU	Soviet Union
DE	Germany	LK	Sri Lanka	TD	Chad
DK	Denmark	LU	Luxembourg	TC	Togo
				US	United States of America

-1-

PARALLEL I/O NETWORK FILE SERVER ARCHITECTURE

5

10 The present application is related to the following  
U.S. Patent Applications, all filed concurrently  
herewith:

15 1. MULTIPLE FACILITY OPERATING SYSTEM  
ARCHITECTURE, invented by David Hitz, Allan Schwartz,  
James Lau and Guy Harris;

2. ENHANCED VMEBUS PROTOCOL UTILIZING  
PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA  
TRANSFER, invented by Daryl Starr; and

20 3. BUS LOCKING FIFO MULTI-PROCESSOR COMMUNICATIONS  
SYSTEM UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND  
BLOCK MODE DATA TRANSFER invented by Daryl D. Starr,  
William Pitts and Stephen Blightman.

25 The above applications are all assigned to the  
assignee of the present invention and are all  
expressly incorporated herein by reference.

**SUBSTITUTE SHEET**

-2-

BACKGROUND OF THE INVENTIONField of the Invention

5 The invention relates to computer data networks, and more particularly, to network file server architectures for computer networks.

Description of the Related Art

10 Over the past ten years, remarkable increases in hardware price/performance ratios have caused a startling shift in both technical and office computing environments. Distributed workstation-server networks are displacing the once pervasive dumb terminal attached to mainframe or minicomputer. To date, however, network I/O limitations have constrained the potential performance available to workstation users. This situation has developed in part because dramatic jumps in microprocessor performance have exceeded increases in network I/O performance.

15 In a computer network, individual user workstations are referred to as clients, and shared resources for filing, printing, data storage and wide-area communications are referred to as servers. Clients and servers are all considered nodes of a network. Client nodes use standard communications protocols to exchange service requests and responses with server nodes.

20 Present-day network clients and servers usually run the DOS, MacIntosh OS, OS/2, or Unix operating systems. Local networks are usually Ethernet or Token Ring at the high end, Arcnet in the midrange, or LocalTalk or StarLAN at the low end. The client-server communication protocols are fairly strictly dictated by the operating system environment -- usually one of several proprietary schemes for PCs (NetWare, 3Plus, Vines, LANManager, LANServer); 25 AppleTalk for MacIntoshes; and TCP/IP with NFS or RFS

**SUBSTITUTE SHEET**

-3-

for Unix. These protocols are all well-known in the industry.

5 Unix client nodes typically feature a 16- or 32-bit microprocessor with 1-8 MB of primary memory, a 640 x 1024 pixel display, and a built-in network interface. A 40-100 MB local disk is often optional. Low-end examples are 80286-based PCs or 68000-based MacIntosh I's; mid-range machines include 80386 PCs, MacIntosh II's, and 680X0-based Unix workstations; 10 high-end machines include RISC-based DEC, HP, and Sun Unix workstations. Servers are typically nothing more than repackaged client nodes, configured in 19-inch racks rather than desk sideboxes. The extra space of a 19-inch rack is used for additional backplane slots, disk or tape drives, and power supplies.

15 Driven by RISC and CISC microprocessor developments, client workstation performance has increased by more than a factor of ten in the last few years. Concurrently, these extremely fast clients 20 have also gained an appetite for data that remote servers are unable to satisfy. Because the I/O shortfall is most dramatic in the Unix environment, the description of the preferred embodiment of the present invention will focus on Unix file servers. 25 The architectural principles that solve the Unix server I/O problem, however, extend easily to server performance bottlenecks in other operating system environments as well. Similarly, the description of the preferred embodiment will focus on Ethernet 30 implementations, though the principles extend easily to other types of networks.

In most Unix environments, clients and servers exchange file data using the Network File System ("NFS"), a standard promulgated by Sun Microsystems 35 and now widely adopted by the Unix community. NFS is defined in a document entitled, "NFS: Network File

**SUBSTITUTE SHEET**

-4-

System Protocol Specification," Request For Comments (RFC) 1094, by Sun Microsystems, Inc. (March 1989). This document is incorporated herein by reference in its entirety.

5           While simple and reliable, NFS is not optimal. Clients using NFS place considerable demands upon both networks and NFS servers supplying clients with NFS data. This demand is particularly acute for so-called diskless clients that have no local disks and  
10           therefore depend on a file server for application binaries and virtual memory paging as well as data. For these Unix client-server configurations, the ten-to-one increase in client power has not been matched by a ten-to-one increase in Ethernet capacity, in disk  
15           speed, or server disk-to-network I/O throughput.

          The result is that the number of diskless clients that a single modern high-end server can adequately support has dropped to between 5-10, depending on client power and application workload. For clients  
20           containing small local disks for applications and paging, referred to as dataless clients, the client-to-server ratio is about twice this, or between 10-20.

          Such low client/server ratios cause piecewise  
25           network configurations in which each local Ethernet contains isolated traffic for its own 5-10 (diskless) clients and dedicated server. For overall connectivity, these local networks are usually joined together with an Ethernet backbone or, in the future,  
30           with an FDDI backbone. These backbones are typically connected to the local networks either by IP routers or MAC-level bridges, coupling the local networks together directly, or by a second server functioning as a network interface, coupling servers for all the  
35           local networks together.

**SUBSTITUTE SHEET**

-5-

In addition to performance considerations, the low client-to-server ratio creates computing problems in several additional ways:

- 5       1. Sharing. Development groups of more than 5-10 people cannot share the same server, and thus cannot easily share files without file replication and manual, multi-server updates. Bridges or routers are a partial solution but inflict a performance penalty due to more network hops.
- 10       2. Administration. System administrators must maintain many limited-capacity servers rather than a few more substantial servers. This burden includes network administration, hardware maintenance, and user account administration.
- 15       3. File System Backup. System administrators or operators must conduct multiple file system backups, which can be onerously time consuming tasks. It is also expensive to duplicate backup peripherals on each server (or every few servers if slower network backup is used).
- 20       4. Price Per Seat. With only 5-10 clients per server, the cost of the server must be shared by only a small number of users. The real cost of an entry-level Unix workstation is therefore significantly greater, often as much as 40% greater, than the cost of the workstation alone.
- 25

The widening I/O gap, as well as administrative and economic considerations, demonstrates a need for higher-performance, larger-capacity Unix file servers. Conversion of a display-less workstation into a server may address disk capacity issues, but does nothing to address fundamental I/O limitations. As an NFS server, the one-time workstation must sustain 5-10 or more times the network, disk, backplane, and file system throughput than it was designed to support as a client. Adding larger disks, more network adaptors,

**SUBSTITUTE SHEET**

-6-

extra primary memory, or even a faster processor do not resolve basic architectural I/O constraints; I/O throughput does not increase sufficiently.

5 Other prior art computer architectures, while not specifically designed as file servers, may potentially be used as such. In one such well-known architecture, a CPU, a memory unit, and two I/O processors are connected to a single bus. One of the I/O processors operates a set of disk drives, and if the architecture  
10 is to be used as a server, the other I/O processor would be connected to a network. This architecture is not optimal as a file server, however, at least because the two I/O processors cannot handle network file requests without involving the CPU. All network  
15 file requests that are received by the network I/O processor are first transmitted to the CPU, which makes appropriate requests to the disk-I/O processor for satisfaction of the network request.

20 In another such computer architecture, a disk controller CPU manages access to disk drives, and several other CPUs, three for example, may be clustered around the disk controller CPU. Each of the other CPUs can be connected to its own network. The network CPUs are each connected to the disk controller  
25 CPU as well as to each other for interprocessor communication. One of the disadvantages of this computer architecture is that each CPU in the system runs its own complete operating system. Thus, network file server requests must be handled by an operating  
30 system which is also heavily loaded with facilities and processes for performing a large number of other, non file-server tasks. Additionally, the interprocessor communication is not optimized for file server type requests.

35 In yet another computer architecture, a plurality of CPUs, each having its own cache memory for data and

**SUBSTITUTE SHEET**



-7-

instruction storage, are connected to a common bus with a system memory and a disk controller. The disk controller and each of the CPUs have direct memory access to the system memory, and one or more of the CPUs can be connected to a network. This architecture is disadvantageous as a file server because, among other things, both file data and the instructions for the CPUs reside in the same system memory. There will be instances, therefore, in which the CPUs must stop running while they wait for large blocks of file data to be transferred between system memory and the network CPU. Additionally, as with both of the previously described computer architectures, the entire operating system runs on each of the CPUs, including the network CPU.

In yet another type of computer architecture, a large number of CPUs are connected together in a hypercube topology. One or more of these CPUs can be connected to networks, while another can be connected to disk drives. This architecture is also disadvantageous as a file server because, among other things, each processor runs the entire operating system. Interprocessor communication is also not optimal for file server applications.

#### 25 SUMMARY OF THE INVENTION

The present invention involves a new, server-specific I/O architecture that is optimized for a Unix file server's most common actions -- file operations. Roughly stated, the invention involves a file server architecture comprising one or more network controllers, one or more file controllers, one or more storage processors, and a system or buffer memory, all connected over a message passing bus and operating in parallel with the Unix host processor. The network controllers each connect to one or more network, and

**SUBSTITUTE SHEET**

-8-

provide all protocol processing between the network layer data format and an internal file server format for communicating client requests to other processors in the server. Only those data packets which cannot  
5 be interpreted by the network controllers, for example client requests to run a client-defined program on the server, are transmitted to the Unix host for processing. Thus the network controllers, file controllers and storage processors contain only small  
10 parts of an overall operating system, and each is optimized for the particular type of work to which it is dedicated.

Client requests for file operations are transmitted to one of the file controllers which, independently of  
15 the Unix host, manages the virtual file system of a mass storage device which is coupled to the storage processors. The file controllers may also control data buffering between the storage processors and the network controllers, through the system memory. The  
20 file controllers preferably each include a local buffer memory for caching file control information, separate from the system memory for caching file data. Additionally, the network controllers, file processors and storage processors are all designed to avoid any  
25 instruction fetches from the system memory, instead keeping all instruction memory separate and local. This arrangement eliminates contention on the backplane between microprocessor instruction fetches and transmissions of message and file data.

30

**BRIEF DESCRIPTION OF THE DRAWINGS**

The invention will be described with respect to particular embodiments thereof, and reference will be made to the drawings, in which:

35

Fig. 1. is a block diagram of a prior art file server architecture;

**SUBSTITUTE SHEET**

-9-

Fig. 2 is a block diagram of a file server architecture according to the invention;

Fig. 3 is a block diagram of one of the network controllers shown in Fig. 2;

5 Fig. 4 is a block diagram of one of the file controllers shown in Fig. 2;

Fig. 5 is a block diagram of one of the storage processors shown in Fig. 2;

10 Fig. 6 is a block diagram of one of the system memory cards shown in Fig. 2;

Figs. 7A-C are a flowchart illustrating the operation of a fast transfer protocol BLOCK WRITE cycle; and

15 Figs. 8A-C are a flowchart illustrating the operation of a fast transfer protocol BLOCK READ cycle.

#### DETAILED DESCRIPTION

20 For comparison purposes and background, an illustrative prior-art file server architecture will first be described with respect to Fig. 1. Fig. 1 is an overall block diagram of a conventional prior-art Unix-based file server for Ethernet networks. It consists of a host CPU card 10 with a single  
25 microprocessor on board. The host CPU card 10 connects to an Ethernet #1 12, and it connects via a memory management unit (MMU) 11 to a large memory array 16. The host CPU card 10 also drives a keyboard, a video display, and two RS232 ports (not  
30 shown). It also connects via the MMU 11 and a standard 32-bit VME bus 20 to various peripheral devices, including an SMD disk controller 22 controlling one or two disk drives 24, a SCSI host adaptor 26 connected to a SCSI bus 28, a tape  
35 controller 30 connected to a quarter-inch tape drive 32, and possibly a network #2 controller 34 connected

**SUBSTITUTE SHEET**

-10-

to a second Ethernet 36. The SMD disk controller 22 can communicate with memory array 16 by direct memory access via bus 20 and MMU 11, with either the disk controller or the MMU acting as a bus master. This configuration is illustrative; many variations are available.

The system communicates over the Ethernets using industry standard TCP/IP and NFS protocol stacks. A description of protocol stacks in general can be found in Tanenbaum, "Computer Networks" (Second Edition, Prentice Hall: 1988). File server protocol stacks are described at pages 535-546. The Tanenbaum reference is incorporated herein by reference.

Basically, the following protocol layers are implemented in the apparatus of Fig. 1:

Network Layer. The network layer converts data packets between a format specific to Ethernets and a format which is independent of the particular type of network used. the Ethernet-specific format which is used in the apparatus of Fig. 1 is described in Hornig, "A Standard For The Transmission of IP Datagrams Over Ethernet Networks," RFC 894 (April 1984), which is incorporated herein by reference.

The Internet Protocol (IP) Layer. This layer provides the functions necessary to deliver a package of bits (an internet datagram) from a source to a destination over an interconnected system of networks. For messages to be sent from the file server to a client, a higher level in the server calls the IP module, providing the internet address of the destination client and the message to transmit. The IP module performs any required fragmentation of the message to accommodate packet size limitations of any intervening gateway, adds internet headers to each fragment, and calls on the network layer to transmit the resulting internet datagrams. The internet header

**SUBSTITUTE SHEET**

-11-

includes a local network destination address (translated from the internet address) as well as other parameters.

5 For messages received by the IP layer from the network layer, the IP module determines from the internet address whether the datagram is to be forwarded to another host on another network, for example on a second Ethernet such as 36 in Fig. 1, or whether it is intended for the server itself. If it  
10 is intended for another host on the second network, the IP module determines a local net address for the destination and calls on the local network layer for that network to send the datagram. If the datagram is intended for an application program within the server,  
15 the IP layer strips off the header and passes the remaining portion of the message to the appropriate next higher layer. The internet protocol standard used in the illustrative apparatus of Fig. 1 is specified in Information Sciences Institute, "Internet Protocol, DARPA Internet Program Protocol Specification," RFC 791 (September 1981), which is  
20 incorporated herein by reference.

TCP/UDP Layer. This layer is a datagram service with more elaborate packaging and addressing options  
25 than the IP layer. For example, whereas an IP datagram can hold about 1,500 bytes and be addressed to hosts, UDP datagrams can hold about 64KB and be addressed to a particular port within a host. TCP and UDP are alternative protocols at this layer;  
30 applications requiring ordered reliable delivery of streams of data may use TCP, whereas applications (such as NFS) which do not require ordered and reliable delivery may use UDP.

The prior art file server of Fig. 1 uses both TCP  
35 and UDP. It uses UDP for file server-related services, and uses TCP for certain other services

**SUBSTITUTE SHEET**

-12-

which the server provides to network clients. The UDP is specified in Postel, "User Datagram Protocol," RFC 768 (August 28, 1980), which is incorporated herein by reference. TCP is specified in Postel, "Transmission Control Protocol," RFC 761 (January 1980) and RFC 793 (September 1981), which is also incorporated herein by reference.

XDR/RPC Layer. This layer provides functions callable from higher level programs to run a designated procedure on a remote machine. It also provides the decoding necessary to permit a client machine to execute a procedure on the server. For example, a caller process in a client node may send a call message to the server of Fig. 1. The call message includes a specification of the desired procedure, and its parameters. The message is passed up the stack to the RPC layer, which calls the appropriate procedure within the server. When the procedure is complete, a reply message is generated and RPC passes it back down the stack and over the network to the caller client. RPC is described in Sun Microsystems, Inc., "RPC: Remote Procedure Call Protocol Specification, Version 2," RFC 1057 (June 1988), which is incorporated herein by reference.

RPC uses the XDR external data representation standard to represent information passed to and from the underlying UDP layer. XDR is merely a data encoding standard, useful for transferring data between different computer architectures. Thus, on the network side of the XDR/RPC layer, information is machine-independent; on the host application side, it may not be. XDR is described in Sun Microsystems, Inc., "XDR: External Data Representation Standard," RFC 1014 (June 1987), which is incorporated herein by reference.

**SUBSTITUTE SHEET**

-13-

NFS Layer. The NFS ("network file system") layer is one of the programs available on the server which an RPC request can call. The combination of host address, program number, and procedure number in an RPC request can specify one remote NFS procedure to be called.

Remote procedure calls to NFS on the file server of Fig. 1 provide transparent, stateless, remote access to shared files on the disks 24. NFS assumes a file system that is hierarchical, with directories as all but the bottom level of files. Client hosts can call any of about 20 NFS procedures including such procedures as reading a specified number of bytes from a specified file; writing a specified number of bytes to a specified file; creating, renaming and removing specified files; parsing directory trees; creating and removing directories; and reading and setting file attributes. The location on disk to which and from which data is stored and retrieved is always specified in logical terms, such as by a file handle or Inode designation and a byte offset. The details of the actual data storage are hidden from the client. The NFS procedures, together with possible higher level modules such as Unix VFS and UFS, perform all conversion of logical data addresses to physical data addresses such as drive, head, track and sector identification. NFS is specified in Sun Microsystems, Inc., "NFS: Network File System Protocol Specification," RFC 1094 (March 1989), incorporated herein by reference.

With the possible exception of the network layer, all the protocol processing described above is done in software, by a single processor in the host CPU card 10. That is, when an Ethernet packet arrives on Ethernet 12, the host CPU 10 performs all the protocol processing in the NFS stack, as well as the protocol

**SUBSTITUTE SHEET**

-14-

processing for any other application which may be running on the host 10. NFS procedures are run on the host CPU 10, with access to memory 16 for both data and program code being provided via MMU 11. Logically  
5 specified data addresses are converted to a much more physically specified form and communicated to the SMD disk controller 22 or the SCSI bus 28, via the VME bus 20, and all disk caching is done by the host CPU 10 through the memory 16. The host CPU card 10 also runs  
10 procedures for performing various other functions of the file server, communicating with tape controller 30 via the VME bus 20. Among these are client-defined remote procedures requested by client workstations.

If the server serves a second Ethernet 36, packets  
15 from that Ethernet are transmitted to the host CPU 10 over the same VME bus 20 in the form of IP datagrams. Again, all protocol processing except for the network layer is performed by software processes running on the host CPU 10. In addition, the protocol processing  
20 for any message that is to be sent from the server out on either of the Ethernets 12 or 36 is also done by processes running on the host CPU 10.

It can be seen that the host CPU 10 performs an enormous amount of processing of data, especially if  
25 5-10 clients on each of the two Ethernets are making file server requests and need to be sent responses on a frequent basis. The host CPU 10 runs a multitasking Unix operating system, so each incoming request need not wait for the previous request to be completely  
30 processed and returned before being processed. Multiple processes are activated on the host CPU 10 for performing different stages of the processing of different requests, so many requests may be in process at the same time. But there is only one CPU on the  
35 card 10, so the processing of these requests is not accomplished in a truly parallel manner. The

**SUBSTITUTE SHEET**



-15-

processes are instead merely time sliced. The CPU 10 therefore represents a major bottleneck in the processing of file server requests.

5 Another bottleneck occurs in MMU 11, which must transmit both instructions and data between the CPU card 10 and the memory 16. All data flowing between the disk drives and the network passes through this interface at least twice.

10 Yet another bottleneck can occur on the VME bus 20, which must transmit data among the SMD disk controller 22, the SCSI host adaptor 26, the host CPU card 10, and possibly the network #2 controller 24.

#### PREFERRED EMBODIMENT-OVERALL HARDWARE ARCHITECTURE

15 In Fig. 2 there is shown a block diagram of a network file server 100 according to the invention. It can include multiple network controller (NC) boards, one or more file controller (FC) boards, one or more storage processor (SP) boards, multiple system  
20 memory boards, and one or more host processors. The particular embodiment shown in Fig. 2 includes four network controller boards 110a-110d, two file controller boards 112a-112b, two storage processors 114a-114b, four system memory cards 116a-116d for a  
25 total of 192MB of memory, and one local host processor 118. The boards 110, 112, 114, 116 and 118 are connected together over a VME bus 120 on which an enhanced block transfer mode as described in the ENHANCED VMEBUS PROTOCOL application identified above  
30 may be used. Each of the four network controllers 110 shown in Fig. 2 can be connected to up to two Ethernets 122, for a total capacity of 8 Ethernets 122a-122h. Each of the storage processors 114 operates ten parallel SCSI busses, nine of which can  
35 each support up to three SCSI disk drives each. The tenth SCSI channel on each of the storage processors

SUBSTITUTE SHEET

-16-

114 is used for tape drives and other SCSI peripherals.

5 The host 118 is essentially a standard SunOs Unix processor, providing all the standard Sun Open Network Computing (ONC) services except NFS and IP routing. Importantly, all network requests to run a user-defined procedure are passed to the host for execution. Each of the NC boards 110, the FC boards 10 112 and the SP boards 114 includes its own independent 32-bit microprocessor. These boards essentially off-load from the host processor 118 virtually all of the NFS and disk processing. Since the vast majority of 15 messages to and from clients over the Ethernets 122 involve NFS requests and responses, the processing of these requests in parallel by the NC, FC and SP processors, with minimal involvement by the local host 118, vastly improves file server performance. Unix is explicitly eliminated from virtually all network, file, and storage processing.

20 OVERALL SOFTWARE ORGANIZATION AND DATA FLOW

Prior to a detailed discussion of the hardware subsystems shown in Fig. 2, an overview of the software structure will now be undertaken. The software organization is described in more detail in 25 the above-identified application entitled MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE.

Most of the elements of the software are well known in the field and are found in most networked Unix systems, but there are two components which are not: 30 Local NFS ("LNFS") and the messaging kernel ("MK") operating system kernel. These two components will be explained first.

The Messaging Kernel. The various processors in file server 100 communicate with each other through 35 the use of a messaging kernel running on each of the

**SUBSTITUTE SHEET**

-17-

processors 110, 112, 114 and 118. These processors do not share any instruction memory, so task-level communication cannot occur via straightforward procedure calls as it does in conventional Unix. Instead, the messaging kernel passes messages over VME bus 120 to accomplish all necessary inter-processor communication. Message passing is preferred over remote procedure calls for reasons of simplicity and speed.

Messages passed by the messaging kernel have a fixed 128-byte length. Within a single processor, messages are sent by reference; between processors, they are copied by the messaging kernel and then delivered to the destination process by reference. The processors of Fig. 2 have special hardware, discussed below, that can expediently exchange and buffer inter-processor messaging kernel messages.

The LNFS Local NFS interface. The 22-function NFS standard was specifically designed for stateless operation using unreliable communication. This means that neither clients nor server can be sure if they hear each other when they talk (unreliability). In practice, an in an Ethernet environment, this works well.

Within the server 100, however, NFS level datagrams are also used for communication between processors, in particular between the network controllers 110 and the file controller 112, and between the host processor 118 and the file controller 112. For this internal communication to be both efficient and convenient, it is undesirable and impractical to have complete statelessness or unreliable communications. Consequently, a modified form of NFS, namely LNFS, is used for internal communication of NFS requests and responses. LNFS is used only within the file server 100; the external network protocol supported by the

**SUBSTITUTE SHEET**

-18-

server is precisely standard, licensed NFS. LNFS is described in more detail below.

5 The Network Controllers 110 each run an NFS server which, after all protocol processing is done up to the NFS layer, converts between external NFS requests and responses and internal LNFS requests and responses. For example, NFS requests arrive as RPC requests with XDR and enclosed in a UDP datagram. After protocol processing, the NFS server translates the NFS request  
10 into LNFS form and uses the messaging kernel to send the request to the file controller 112.

The file controller runs an LNFS server which handles LNFS requests both from network controllers and from the host 118. The LNFS server translates  
15 LNFS requests to a form appropriate for a file system server, also running on the file controller, which manages the system memory file data cache through a block I/O layer.

20 An overview of the software in each of the processors will now be set forth.

#### Network Controller 110

25 The optimized dataflow of the server 100 begins with the intelligent network controller 110. This processor receives Ethernet packets from client workstations. It quickly identifies NFS-destined packets and then performs full protocol processing on them to the NFS level, passing the resulting LNFS requests directly to the file controller 112. This  
30 protocol processing includes IP routing and reassembly, UDP demultiplexing, XDR decoding, and NFS request dispatching. The reverse steps are used to send an NFS reply back to a client. Importantly, these time-consuming activities are performed directly  
35 in the Network Controller 110, not in the host 118.

**SUBSTITUTE SHEET**

-19-

The server 100 uses conventional NFS ported from Sun Microsystems, Inc., Mountain View, CA, and is NFS protocol compatible.

5 Non-NFS network traffic is passed directly to its destination host processor 118.

The NCs 110 also perform their own IP routing. Each network controller 110 supports two fully parallel Ethernets. There are four network controllers in the embodiment of the server 100 shown in Fig. 2, so that server can support up to eight Ethernets. For the two Ethernets on the same network controller 110, IP routing occurs completely within the network controller and generates no backplane traffic. Thus attaching two mutually active Ethernets to the same controller not only minimizes their inter-  
10 net transit time, but also significantly reduces backplane contention on the VME bus 120. Routing table updates are distributed to the network controllers from the host processor 118, which runs  
15 either the gated or routed Unix demon.  
20

While the network controller described here is designed for Ethernet LANs, it will be understood that the invention can be used just as readily with other network types, including FDDI.

25 File Controller 112

In addition to dedicating a separate processor for NFS protocol processing and IP routing, the server 100 also dedicates a separate processor, the intelligent file controller 112, to be responsible for all file  
30 system processing. It uses conventional Berkeley Unix 4.3 file system code and uses a binary-compatible data representation on disk. These two choices allow all standard file system utilities (particularly block-level tools) to run unchanged.

**SUBSTITUTE SHEET**

-20-

The file controller 112 runs the shared file system used by all NCs 110 and the host processor 118. Both the NCs and the host processor communicate with the file controller 112 using the LNFS interface. The NCs  
5 110 use LNFS as described above, while the host processor 118 uses LNFS as a plug-in module to SunOs's standard Virtual File System ("VFS") interface.

When an NC receives an NFS read request from a client workstation, the resulting LNFS request passes  
10 to the FC 112. The FC 112 first searches the system memory 116 buffer cache for the requested data. If found, a reference to the buffer is returned to the NC 110. If not found, the LRU (least recently used) cache buffer in system memory 116 is freed and  
15 reassigned for the requested block. The FC then directs the SP 114 to read the block into the cache buffer from a disk drive array. When complete, the SP so notifies the FC, which in turn notifies the NC 100. The NC 110 then sends an NFS reply, with the data from  
20 the buffer, back to the NFS client workstation out on the network. Note that the SP 114 transfers the data into system memory 116, if necessary, and the NC 110 transferred the data from system memory 116 to the networks. The process takes place without any  
25 involvement of the host 118.

#### Storage Processor

The intelligent storage processor 114 manages all disk and tape storage operations. While autonomous,  
30 storage processors are primarily directed by the file controller 112 to move file data between system memory 116 and the disk subsystem. The exclusion of both the host 118 and the FC 112 from the actual data path helps to supply the performance needed to service many  
35 remote clients.

**SUBSTITUTE SHEET**

-21-

5           Additionally, coordinated by a Server Manager in  
the host 118, storage processor 114 can execute server  
backup by moving data between the disk subsystem and  
tape or other archival peripherals on the SCSI  
channels. Further, if directly accessed by host  
processor 118, SP 114 can provide a much higher  
performance conventional disk interface for Unix,  
virtual memory, and databases. In Unix nomenclature,  
the host processor 118 can mount boot, storage swap,  
10           and raw partitions via the storage processors 114.

Each storage processor 114 operates ten parallel,  
fully synchronous SCSI channels (busses)  
simultaneously. Nine of these channels support three  
arrays of nine SCSI disk drives each, each drive in an  
array being assigned to a different SCSI channel. The  
15           tenth SCSI channel hosts up to seven tape and other  
SCSI peripherals. In addition to performing reads and  
writes, SP 114 performs device-level optimizations  
such as disk seek queue sorting, directs device error  
recovery, and controls DMA transfers between the  
20           devices and system memory 116.

#### Host Processor 118

25           The local host 118 has three main purposes: to run  
Unix, to provide standard ONC network services for  
clients, and to run a Server Manager. Since Unix and  
ONC are ported from the standard SunOs Release 4 and  
ONC Services Release 2, the server 100 can provide  
identically compatible high-level ONC services such as  
30           the Yellow Pages, Lock Manager, DES Key Authenticator,  
Auto Mounter, and Port Mapper. Sun/2 Network disk  
booting and more general IP internet services such as  
Telnet, FTP, SMTP, SNMP, and reverse ARP are also  
supported. Finally, print spoolers and similar Unix  
35           demons operate transparently.

**SUBSTITUTE SHEET**

-22-

The host processor 118 runs the following software modules:

5        TCP and socket layers. The Transport Control Protocol ("TCP"), which is used for certain server functions other than NFS, provides reliable bytestream communication between two processors. Socket are used to establish TCP connections.

10       VFS interface. The Virtual File System ("VFS") interface is a standard SunOs file system interface. It paints a uniform file-system picture for both users and the non-file parts of the Unix operating system, hiding the details of the specific file system. Thus standard NFS, LNFS, and any local Unix file system can coexist harmoniously.

15       UFS interface. The Unix File System ("UFS") interface is the traditional and well-known Unix interface for communication with local-to-the-processor disk drives. In the server 100, it is used to occasionally mount storage processor volumes directly, without going through the file controller 20 112. Normally, the host 118 uses LNFS and goes through the file controller.

25       Device layer. The device layer is a standard software interface between the Unix device model and different physical device implementations. In the server 100, disk devices are not attached to host processors directly, so the disk driver in the host's device layer uses the messaging kernel to communicate with the storage processor 114.

30       Route and Port Mapper Demons. The Route and Port Mapper demons are Unix user-level background processes that maintain the Route and Port databases for packet routing. They are mostly inactive and not in any performance path.

35       Yellow Pages and Authentication Demon. The Yellow Pages and Authentication services are Sun-ONC standard

**SUBSTITUTE SHEET**



-23-

network services. Yellow Pages is a widely used multipurpose name-to-name directory lookup service. The Authentication service uses cryptographic keys to authenticate, or validate, requests to insure that requestors have the proper privileges for any actions or data they desire.

Server Manager. The Server Manager is an administrative application suite that controls configuration, logs error and performance reports, and provides a monitoring and tuning interface for the system administrator. These functions can be exercised from either system console connected to the host 118, or from a system administrator's workstation.

The host processor 118 is a conventional OEM Sun central processor card, Model 3E/120. It incorporates a Motorola 68020 microprocessor and 4MB of on-board memory. Other processors, such as a SPARC-based processor, are also possible.

The structure and operation of each of the hardware components of server 100 will now be described in detail.

#### NETWORK CONTROLLER HARDWARE ARCHITECTURE

Fig. 3 is a block diagram showing the data path and some control paths for an illustrative one of the network controllers 110a. It comprises a 20 MHz 68020 microprocessor 210 connected to a 32-bit microprocessor data bus 212. Also connected to the microprocessor data bus 212 is a 256K byte CPU memory 214. The low order 8 bits of the microprocessor data bus 212 are connected through a bidirectional buffer 216 to an 8-bit slow-speed data bus 218. On the slow-speed data bus 218 is a 128K byte EPROM 220, a 32 byte PROM 222, and a multi-function peripheral (MFP) 224. The EPROM 220 contains boot code for the network

**SUBSTITUTE SHEET**

-24-

controller 110a, while the PROM 222 stores various operating parameters such as the Ethernet addresses assigned to each of the two Ethernet interfaces on the board. Ethernet address information is read into the  
5 corresponding interface control block in the CPU memory 214 during initialization. The MFP 224 is a Motorola 68901, and performs various local functions such as timing, interrupts, and general purpose I/O. The MFP 224 also includes a UART for interfacing to an  
10 RS232 port 226. These functions are not critical to the invention and will not be further described herein.

The low order 16 bits of the microprocessor data bus 212 are also coupled through a bidirectional  
15 buffer 230 to a 16-bit LAN data bus 232. A LAN controller chip 234, such as the Am7990 LANCE Ethernet controller manufactured by Advanced Micro Devices, Inc. Sunnyvale, CA., interfaces the LAN data bus 232 with the first Ethernet 122a shown in Fig. 2. Control and data for the LAN controller 234 are stored in a  
20 512K byte LAN memory 236, which is also connected to the LAN data bus 232. A specialized 16 to 32 bit FIFO chip 240, referred to herein as a parity FIFO chip and described below, is also connected to the LAN data bus  
25 232. Also connected to the LAN data bus 232 is a LAN DMA controller 242, which controls movements of packets of data between the LAN memory 236 and the FIFO chip 240. The LAN DMA controller 242 may be a Motorola M68440 DMA controller using channel zero  
30 only.

The second Ethernet 122b shown in Fig. 2 connects to a second LAN data bus 252 on the network controller  
card 110a shown in Fig. 3. The LAN data bus 252 connects to the low order 16 bits of the  
35 microprocessor data bus 212 via a bidirectional buffer 250, and has similar components to those appearing on

**SUBSTITUTE SHEET**

-25-

5 the LAN data bus 232. In particular, a LAN controller 254 interfaces the LAN data bus 252 with the Ethernet 122b, using LAN memory 256 for data and control, and a LAN DMA controller 262 controls DMA transfer of data between the LAN memory 256 and the 16-bit wide data port A of the parity FIFO 260.

10 The low order 16 bits of microprocessor data bus 212 are also connected directly to another parity FIFO 270, and also to a control port of a VME/FIFO DMA controller 272. The FIFO 270 is used for passing messages between the CPU memory 214 and one of the remote boards 110, 112, 114, 116 or 118 (Fig. 2) in a manner described below. The VME/FIFO DMA controller 272, which supports three round-robin non-prioritized channels for copying data, controls all data transfers  
15 between one of the remote boards and any of the FIFOs 240, 260 or 270, as well as between the FIFOs 240 and 260.

20 32-bit data bus 274, which is connected to the 32-bit port B of each of the FIFOs 240, 260 and 270, is the data bus over which these transfers take place. Data bus 274 communicates with a local 32-bit bus 276 via a bidirectional pipelining latch 278, which is also controlled by VME/FIFO DMA controller 727, which  
25 in turn communicates with the VME bus 120 via a bidirectional buffer 280.

The local data bus 276 is also connected to a set of control registers 282, which are directly addressable across the VME bus 120. The registers 282  
30 are used mostly for system initialization and diagnostics.

The local data bus 276 is also coupled to the microprocessor data bus 212 via a bidirectional buffer 284. When the NC 110a operates in slave mode, the CPU memory 214 is directly addressable from VME bus 120.  
35 One of the remote boards can copy data directly from

**SUBSTITUTE SHEET**

the CPU memory 214 via the bidirectional buffer 284. LAN memories 236 and 256 are not directly addressed over VME bus 120.

5 The parity FIFOs 240, 260 and 270 each consist of an ASIC, the functions and operation of which are described in the Appendix. The FIFOs 240 and 260 are configured for packet data transfer and the FIFO 270 is configured for message passing. Referring to the Appendix, the FIFOs 240 and 260 are programmed with  
 10 the following bit settings in the Data Transfer Configuration Register:

Bit	Definition	Setting
0	WD Mode	N/A
1	Parity Chip	N/A
15 2	Parity Correct Mode	N/A
3	8/16 bits CPU & PortA interface	16 bits (1)
4	Invert Port A address 0	no (0)
5	Invert Port A address 1	yes (1)
6	Checksum Carry Wrap	yes (1)
20 7	Reset	no (0)

The Data Transfer Control Register is programmed as follows:

Bit	Definition	Setting
0	Enable PortA Req/Ack	yes (1)
25 1	Enable PortB Req/Ack	yes (1)
2	Data Transfer Direction	(as desired)
3	CPU parity enable	no (0)
4	PortA parity enable	no (0)
5	PortB parity enable	no (0)
30 6	Checksum Enable	yes (1)
7	PortA Master	yes (1)

Unlike the configuration used on FIFOs 240 and 260, the microprocessor 210 is responsible for loading and unloading Port A directly. The microprocessor 210  
 35 reads an entire 32-bit word from port A with a single instruction using two port A access cycles. Port A

**SUBSTITUTE SHEET**

-27-

data transfer is disabled by unsetting bits 0 (Enable PortA Req/Ack) and 7 (PortA Master) of the Data Transfer Control Register.

5 The remainder of the control settings in FIFO 270 are the same as those in FIFOs 240 and 260 described above.

10 The NC 110a also includes a command FIFO 290. The command FIFO 290 includes an input port coupled to the local data bus 276, and which is directly addressable across the VME bus 120, and includes an output port connected to the microprocessor data bus 212. As explained in more detail below, when one of the remote boards issues a command or response to the NC 110a, it does so by directly writing a 1-word (32-bit) message descriptor into NC 110a's command FIFO 290. Command FIFO 290 generates a "FIFO not empty" status to the microprocessor 210, which then reads the message descriptor off the top of FIFO 290 and processes it. If the message is a command, then it includes a VME address at which the message is located (presumably an address in a shared memory similar to 214 on one of the remote boards). The microprocessor 210 then programs the FIFO 270 and the VME/FIFO DMA controller 272 to copy the message from the remote location into the CPU memory 214.

25 Command FIFO 290 is a conventional two-port FIFO, except that additional circuitry is included for generating a Bus Error signal on VME bus 120 if an attempt is made to write to the data input port while the FIFO is full. Command FIFO 290 has space for 256 entries.

30 A noteworthy feature of the architecture of NC 110a is that the LAN buses 232 and 252 are independent of the microprocessor data bus 212. Data packets being routed to or from an Ethernet are stored in LAN memory 35 236 on the LAN data bus 232 (or 256 on the LAN data

**SUBSTITUTE SHEET**

-28-

bus 252), and not in the CPU memory 214. Data transfer between the LAN memories 236 and 256 and the Ethernets 122a and 122b, are controlled by LAN controllers 234 and 254, respectively, while most data transfer between LAN memory 236 or 256 and a remote port on the VME bus 120 are controlled by LAN DMA controllers 242 and 262, FIFOs 240 and 260, and VME/FIFO DMA controller 272. An exception to this rule occurs when the size of the data transfer is small, e.g., less than 64 bytes, in which case microprocessor 210 copies it directly without using DMA. The microprocessor 210 is not involved in larger transfers except in initiating them and in receiving notification when they are complete.

The CPU memory 214 contains mostly instructions for microprocessor 210, messages being transmitted to or from a remote board via FIFO 270, and various data blocks for controlling the FIFOs, the DMA controllers and the LAN controllers. The microprocessor 210 accesses the data packets in the LAN memories 236 and 256 by directly addressing them through the bidirectional buffers 230 and 250, respectively, for protocol processing. The local high-speed static RAM in CPU memory 214 can therefore provide zero wait state memory access for microprocessor 210 independent of network traffic. This is in sharp contrast to the prior art architecture shown in Fig. 1, in which all data and data packets, as well as microprocessor instructions for host CPU card 10, reside in the memory 16 and must communicate with the host CPU card 10 via the MMU 11.

While the LAN data buses 232 and 252 are shown as separate buses in Fig. 3, it will be understood that they may instead be implemented as a single combined bus.

**SUBSTITUTE SHEET**

-29-

NETWORK CONTROLLER OPERATION

5 In operation, when one of the LAN controllers (such  
as 234) receives a packet of information over its  
Ethernet 122a, it reads in the entire packet and  
stores it in corresponding LAN memory 236. The LAN  
controller 234 then issues an interrupt to  
microprocessor 210 via MFP 224, and the microprocessor  
210 examines the status register on LAN controller 234  
10 (via bidirectional buffer 230) to determine that the  
event causing the interrupt was a "receive packet  
completed." In order to avoid a potential lockout of  
the second Ethernet 122b caused by the prioritized  
interrupt handling characteristic of MFP 224, the  
microprocessor 210 does not at this time immediately  
15 process the received packet; instead, such processing  
is scheduled for a polling function.

When the polling function reaches the processing of  
the received packet, control over the packet is passed  
to a software link level receive module. The link  
20 level receive module then decodes the packet according  
to either of two different frame formats: standard  
Ethernet format or SNAP (IEEE 802 LCC) format. An  
entry in the header in the packet specifies which  
frame format was used. The link level driver then  
25 determines which of three types of messages is  
contained in the received packet: (1) IP, (2) ARP  
packets which can be handled by a local ARP module, or  
(3) ARP packets and other packet types which must be  
forwarded to the local host 118 (Fig. 2) for  
30 processing. If the packet is an ARP packet which can  
be handled by the NC 110a, such as a request for the  
address of server 100, then the microprocessor 210  
assembles a response packet in LAN memory 236 and, in  
a conventional manner, causes LAN controller 234 to  
35 transmit that packet back over Ethernet 122a. It is  
noteworthy that the data manipulation for

**SUBSTITUTE SHEET**

-30-

accomplishing this task is performed almost completely in LAN memory 236, directly addressed by microprocessor 210 as controlled by instructions in CPU memory 214. The function is accomplished also  
5 without generating any traffic on the VME backplane 120 at all, and without disturbing the local host 118.

If the received packet is either an ARP packet which cannot be processed completely in the NC 110a, or is another type of packet which requires delivery  
10 to the local host 118 (such as a client request for the server 100 to execute a client-defined procedure), then the microprocessor 210 programs LAN DMA controller 242 to load the packet from LAN memory 236 into FIFO 240, programs FIFO 240 with the direction of  
15 data transfer, and programs DMA controller 272 to read the packet out of FIFO 240 and across the VME bus 120 into system memory 116. In particular, the microprocessor 210 first programs the LAN DMA controller 242 with the starting address and length of  
20 the packet in LAN memory 236, and programs the controller to begin transferring data from the LAN memory 236 to port A of parity FIFO 240 as soon as the FIFO is ready to receive data. Second, microprocessor 210 programs the VME/FIFO DMA controller 272 with the  
25 destination address in system memory 116 and the length of the data packet, and instructs the controller to begin transferring data from port B of the FIFO 260 onto VME bus 120. Finally, the microprocessor 210 programs FIFO 240 with the  
30 direction of the transfer to take place. The transfer then proceeds entirely under the control of DMA controllers 242 and 272, without any further involvement by microprocessor 210.

The microprocessor 210 then sends a message to host  
35 118 that a packet is available at a specified system memory address. The microprocessor 210 sends such a

**SUBSTITUTE SHEET**



-31-

message by writing a message descriptor to a software-emulated command FIFO on the host, which copies the message from CPU memory 214 on the NC via buffer 284 and into the host's local memory, in ordinary VME block transfer mode. The host then copies the packet from system memory 116 into the host's own local memory using ordinary VME transfers.

If the packet received by NC 110a from the network is an IP packet, then the microprocessor 210 determines whether it is (1) an IP packet for the server 100 which is not an NFS packet; (2) an IP packet to be routed to a different network; or (3) an NFS packet. If it is an IP packet for the server 100, but not an NFS packet, then the microprocessor 210 causes the packet to be transmitted from the LAN memory 236 to the host 118 in the same manner described above with respect to certain ARP packets.

If the IP packet is not intended for the server 100, but rather is to be routed to a client on a different network, then the packet is copied into the LAN memory associated with the Ethernet to which the destination client is connected. If the destination client is on the Ethernet 122b, which is on the same NC board as the source Ethernet 122a, then the microprocessor 210 causes the packet to be copied from LAN memory 236 into LAN 256 and then causes LAN controller 254 to transmit it over Ethernet 122b. (Of course, if the two LAN data buses 232 and 252 are combined, then copying would be unnecessary; the microprocessor 210 would simply cause the LAN controller 254 to read the packet out of the same locations in LAN memory to which the packet was written by LAN controller 234.)

The copying of a packet from LAN memory 236 to LAN memory 256 takes place similarly to the copying described above from LAN memory to system memory. For

**SUBSTITUTE SHEET**

-32-

transfer sizes of 64 bytes or more, the microprocessor 210 first programs the LAN DMA controller 242 with the starting address and length of the packet in LAN memory 236, and programs the controller to begin  
5 transferring data from the LAN memory 236 into port A of parity FIFO 240 as soon as the FIFO is ready to receive data. Second, microprocessor 210 programs the LAN DMA controller 262 with a destination address in LAN memory 256 and the length of the data packet, and  
10 instructs that controller to transfer data from parity FIFO 260 into the LAN memory 256. Third, microprocessor 210 programs the VME/FIFO DMA controller 272 to clock words of data out of port B of the FIFO 240, over the data bus 274, and into port B  
15 of FIFO 260. Finally, the microprocessor 210 programs the two FIFOs 240 and 260 with the direction of the transfer to take place. The transfer then proceeds entirely under the control of DMA controllers 242, 262 and 272, without any further involvement by the  
20 microprocessor 210. Like the copying from LAN memory to system memory, if the transfer size is smaller than 64 bytes, the microprocessor 210 performs the transfer directly, without DMA.

When each of the LAN DMA controllers 242 and 262  
25 complete their work, they so notify microprocessor 210 by a respective interrupt provided through MFP 224. When the microprocessor 210 has received both interrupts, it programs LAN controller 254 to transmit the packet on the Ethernet 122b in a conventional  
30 manner.

Thus, IP routing between the two Ethernets in a single network controller 110 takes place over data bus 274, generating no traffic over VME bus 120. Nor is the host processor 118 disturbed for such routing,  
35 in contrast to the prior art architecture of Fig. 1. Moreover, all but the shortest copying work is

**SUBSTITUTE SHEET**

-33-

5 performed by controllers outside microprocessor 210,  
requiring the involvement of the microprocessor 210,  
and bus traffic on microprocessor data bus 212, only  
for the supervisory functions of programming the DMA  
controllers and the parity FIFOs and instructing them  
to begin. The VME/FIFO DMA controller 272 is  
programmed by loading control registers via  
microprocessor data bus 212; the LAN DMA controllers  
242 and 262 are programmed by loading control  
10 registers on the respective controllers via the  
microprocessor data bus 212, respective bidirectional  
buffers 230 and 250, and respective LAN data buses 232  
and 252, and the parity FIFOs 240 and 260 are  
programmed as set forth in the Appendix.

15 If the destination workstation of the IP packet to  
be routed is on an Ethernet connected to a different  
one of the network controllers 110, then the packet is  
copied into the appropriate LAN memory on the NC 110  
to which that Ethernet is connected. Such copying is  
20 accomplished by first copying the packet into system  
memory 116, in the manner described above with respect  
to certain ARP packets, and then notifying the  
destination NC that a packet is available. When an NC  
is so notified, it programs its own parity FIFO and  
25 DMA controllers to copy the packet from system memory  
116 into the appropriate LAN memory. It is noteworthy  
that though this type of IP routing does create VME  
bus traffic, it still does not involve the host CPU  
118.

30 If the IP packet received over the Ethernet 122a  
and now stored in LAN memory 236 is an NFS packet  
intended for the server 100, then the microprocessor  
210 performs all necessary protocol preprocessing to  
extract the NFS message and convert it to the local  
35 NFS (LNFS) format. This may well involve the logical  
concatenation of data extracted from a large number of

**SUBSTITUTE SHEET**

-34-

individual IP packets stored in LAN memory 236, resulting in a linked list, in CPU memory 214, pointing to the different blocks of data in LAN memory 236 in the correct sequence.

5       The exact details of the LNFS format are not important for an understanding of the invention, except to note that it includes commands to maintain a directory of files which are stored on the disks attached to the storage processors 114, commands for  
10       reading and writing data to and from a file on the disks, and various configuration management and diagnostics control messages. The directory maintenance commands which are supported by LNFS include the following messages based on conventional  
15       NFS: get attributes of a file (GETATTR); set attributes of a file (SETATTR); look up a file (LOOKUP); create a file (CREATE); remove a file (REMOVE); rename a file (RENAME); create a new linked file (LINK); create a symlink (SYMLINK); remove a  
20       directory (RMDIR); and return file system statistics (STATFS). The data transfer commands supported by LNFS include read from a file (READ); write to a file (WRITE); read from a directory (REaddir); and read a link (READLINK). LNFS also supports a buffer release  
25       command (RELEASE), for notifying the file controller that an NC is finished using a specified buffer in system memory. It also supports a VOP-derived access command, for determining whether a given type access is legal for specified credential on a specified file.  
30       If the LNFS request includes the writing of file data from the LAN memory 236 to disk, the NC 110a first requests a buffer in system memory 116 to be allocated by the appropriate FC 112. When a pointer to the buffer is returned, microprocessor 210 programs  
35       LAN DMA controller 242, parity FIFO 240 and VME/FIFO DMA controller 272 to transmit the entire block of

**SUBSTITUTE SHEET**

-35-

file data to system memory 116. The only difference between this transfer and the transfer described above for transmitting IP packets and ARP packets to system memory 116 is that these data blocks will typically have portions scattered throughout LAN memory 236. The microprocessor 210 accommodates that situation by programming LAN DMA controller 242 successively for each portion of the data, in accordance with the linked list, after receiving notification that the previous portion is complete. The microprocessor 210 can program the parity FIFO 240 and the VME/FIFO DMA controller 272 once for the entire message, as long as the entire data block is to be placed contiguously in system memory 116. If it is not, then the microprocessor 210 can program the DMA controller 272 for successive blocks in the same manner LAN DMA controller 242.

If the network controller 110a receives a message from another processor in server 100, usually from file controller 112, that file data is available in system memory 116 for transmission on one of the Ethernets, for example Ethernet 122a, then the network controller 110a copies the file data into LAN memory 236 in a manner similar to the copying of file data in the opposite direction. In particular, the microprocessor 210 first programs VME/FIFO DMA controller 272 with the starting address and length of the data in system memory 116, and programs the controller to begin transferring data over the VME bus 120 into port B of parity FIFO 240 as soon as the FIFO is ready to receive data. The microprocessor 210 then programs the LAN DMA controller 242 with a destination address in LAN memory 236 and then length of the file data, and instructs that controller to transfer data from the parity FIFO 240 into the LAN memory 236. Third, microprocessor 210 programs the parity FIFO 240

**SUBSTITUTE SHEET**

-36-

with the direction of the transfer to take place. The transfer then proceeds entirely under the control of DMA controllers 242 and 272, without any further involvement by the microprocessor 210. Again, if the  
5 file data is scattered in multiple blocks in system memory 116, the microprocessor 210 programs the VME/FIFO DMA controller 272 with a linked list of the blocks to transfer in the proper order.

When each of the DMA controllers 242 and 272  
10 complete their work, they so notify microprocessor 210 through MFP 224. The microprocessor 210 then performs all necessary protocol processing on the LNFS message in LAN memory 236 in order to prepare the message for transmission over the Ethernet 122a in the form of  
15 Ethernet IP packets. As set forth above, this protocol processing is performed entirely in network controller 110a, without any involvement of the local host 118.

It should be noted that the parity FIFOs are  
20 designed to move multiples of 128-byte blocks most efficiently. The data transfer size through port B is always 32-bits wide, and the VME address corresponding to the 32-bit data must be quad-byte aligned. The data transfer size for port A can be either 8 or 16  
25 bits. For bus utilization reasons, it is set to 16 bits when the corresponding local start address is double-byte aligned, and is set at 8 bits otherwise. The TCP/IP checksum is always computed in the 16 bit mode. Therefore, the checksum word requires byte  
30 swapping if the local start address is not double-byte aligned.

Accordingly, for transfer from port B to port A of  
any of the FIFOs 240, 260 or 270, the microprocessor 210 programs the VME/FIFO DMA controller to pad the  
35 transfer count to the next 128-byte boundary. The extra 32-bit word transfers do not involve the VME

**SUBSTITUTE SHEET**

-37-

bus, and only the desired number of 32-bit words will be unloaded from port A.

5 For transfers from port A to port B of the parity FIFO 270, the microprocessor 210 loads port A word-by-word and forces a FIFO full indication when it is finished. The FIFO full indication enables unloading from port B. The same procedure also takes place for  
10 transfers from port A to port B of either of the parity FIFOs 240 or 260, since transfers of fewer than 128 bytes are performed under local microprocessor control rather than under the control of LAN DMA controller 242 or 262. For all of the FIFOs, the VME/FIFO DMA controller is programmed to unload only the desired number of 32-bit words.

#### 15 FILE CONTROLLER HARDWARE ARCHITECTURE

The file controllers (FC) 112 may each be a standard off-the-shelf microprocessor board, such as one manufactured by Motorola Inc. Preferably, however, a more specialized board is used such as that  
20 shown in block diagram form in Fig. 4.

Fig. 4 shows one of the FCs 112a, and it will be understood that the other FC can be identical. In many aspects it is simply a scaled-down version of the NC 110a shown in Fig. 3, and in some respects it is  
25 scaled up. Like the NC 110a, FC 112a comprises a 20MHz 68020 microprocessor 310 connected to a 32-bit microprocessor data bus 312. Also connected to the microprocessor data bus 312 is a 256K byte shared CPU memory 314. The low order 8 bits of the  
30 microprocessor data bus 312 are connected through a bidirectional buffer 316 to an 8-bit slow-speed data bus 318. On slow-speed data bus 318 are a 128K byte PROM 320, and a multifunction peripheral (MFP) 324. The functions of the PROM 320 and MFP 324 are the same  
35 as those described above with respect to EPROM 220 and

**SUBSTITUTE SHEET**

-38-

MFP 224 on NC 110a. FC 112a does not include PROM like the PROM 222 on NC 110a, but does include a parallel port 392. The parallel port 392 is mainly for testing and diagnostics.

5        Like the NC 110a, the FC 112a is connected to the VME bus 120 via a bidirectional buffer 380 and a 32-bit local data bus 376. A set of control registers 382 are connected to the local data bus 376, and directly addressable across the VME bus 120. The  
10       local data bus 376 is also coupled to the microprocessor data bus 312 via a bidirectional buffer 384. This permits the direct addressability of CPU memory 314 from VME bus 120.

15       FC 112a also includes a command FIFO 390, which includes an input port coupled to the local data bus 376 and which is directly addressable across the VME bus 120. The command FIFO 390 also includes an output port connected to the microprocessor data bus 312. The structure, operation and purpose of command FIFO  
20       390 are the same as those described above with respect to command FIFO 290 on NC 110a.

25       The FC 112a omits the LAN data buses 323 and 352 which are present in NC 110a, but instead includes a 4 megabyte 32-bit wide FC memory 396 coupled to the microprocessor data bus 312 via a bidirectional buffer 394. As will be seen, FC memory 396 is used as a cache memory for file control information, separate from the file data information cached in system memory  
30       116.

35       The file controller embodiment shown in Fig. 4 does not include any DMA controllers, and hence cannot act as a master for transmitting or receiving data in any block transfer mode, over the VME bus 120. Block transfers do occur with the CPU memory 314 and the FC memory 396, however, with the FC 112a acting as an VME bus slave. In such transfers, the remote master

**SUBSTITUTE SHEET**



-39-

addresses the CPU memory 314 or the FC memory 396 directly over the VME bus 120 through the bidirectional buffers 384 and, if appropriate, 394.

5     FILE CONTROLLER OPERATION

The purpose of the FC 112a is basically to provide virtual file system services in response to requests provided in LNFS format by remote processors on the VME bus 120. Most requests will come from a network controller 110, but requests may also come from the local host 118.

10     The file related commands supported by LNFS are identified above. They are all specified to the FC 112a in terms of logically identified disk data blocks. For example, the LNFS command for reading data from a file includes a specification of the file from which to read (file system ID (FSID) and file ID (inode)), a byte offset, and a count of the number of bytes to read. The FC 112a converts that identification into physical form, namely disk and sector numbers, in order to satisfy the command.

15     The FC 112a runs a conventional Fast File System (FFS or UFS), which is based on the Berkeley 4.3 VAX release. This code performs the conversion and also performs all disk data caching and control data caching. However, as previously mentioned, control data caching is performed using the FC memory 396 on FC 112a, whereas disk data caching is performed using the system memory 116 (Fig. 2). Caching this file control information within the FC 112a avoids the VME bus congestion and speed degradation which would result if file control information was cached in system memory 116. The memory on the FC 112a is directly accessed over the VME bus 120 for three main purposes. First, and by far the most frequent, are accesses to FC memory 396 by an SP 114 to read or

**SUBSTITUTE SHEET**

-40-

write cached file control information. These are accesses requested by FC 112a to write locally modified file control structures through to disk, or to read file control structures from disk. Second, the FC's CPU memory 314 is accessed directly by other processors for message transmissions from the FC 112a to such other processors. For example, if a data block in system memory is to be transferred to an SP 114 for writing to disk, the FC 112a first assembles a message in its local memory 314 requesting such a transfer. The FC 112a then notifies the SP 114, which copies the message directly from the CPU memory 314 and executes the requested transfer.

A third type of direct access to the FC's local memory occurs when an LNFS client reads directory entries. When FC 112a receives an LNFS request to read directory entries, the FC 112a formats the requested directory entries in FC memory 396 and notifies the requestor of their location. The requestor then directly accesses FC memory 396 to read the entries.

The version of the UFS code on FC 112a includes some modifications in order to separate the two caches. In particular, two sets of buffer headers are maintained, one for the FC memory 396 and one for the system memory 116. Additionally, a second set of the system buffer routines (GETBLK(), BRELSE(), BREAD(), BWRITE(), and BREADA()) exist, one for buffer accesses to FC Mem 396 and one for buffer accesses to system memory 116. The UFS code is further modified to call the appropriate buffer routines for FC memory 396 for accesses to file control information, and to call the appropriate buffer routines for the system memory 116 for the caching of disk data. A description of UFS may be found in chapters 2, 6, 7 and 8 of "Kernel Structure and Flow," by Rieken and Webb of .sh

**SUBSTITUTE SHEET**

-41-

consulting (Santa Clara, California: 1988), incorporated herein by reference.

5 When a read command is sent to the FC by a requestor such as a network controller, the FC first converts the file, offset and count information into disk and sector information. It then locks the system memory buffers which contain that information, instructing the storage processor 114 to read them from disk if necessary. When the buffer is ready, the 10 FC returns a message to the requestor containing both the attributes of the designated file and an array of buffer descriptors that identify the locations in system memory 116 holding the data.

15 After the requestor has read the data out of the buffers, it sends a release request back to the FC. The release request is the same message that was returned by the FC in response to the read request; the FC 112a uses the information contained therein to determine which buffers to free.

20 A write command is processed by FC 112a similarly to the read command, but the caller is expected to write to (instead of read from) the locations in system memory 116 identified by the buffer descriptors returned by the FC 112a. Since FC 112a employs write-through caching, when it receives the release command 25 from the requestor, it instructs storage processor 114 to copy the data from system memory 116 onto the appropriate disk sectors before freeing the system memory buffers for possible reallocation.

30 The READDIR transaction is similar to read and write, but the request is satisfied by the FC 112a directly out of its own FC memory 396 after formatting the requested directory information specifically for this purpose. The FC 112a causes the storage processor read the requested directory information 35 from disk if it is not already locally cached. Also,

**SUBSTITUTE SHEET**

-42-

the specified offset is a "magic cookie" instead of a byte offset, identifying directory entries instead of an absolute byte offset into the file. No file attributes are returned.

5       The READLINK transaction also returns no file attributes, and since links are always read in their entirety, it does not require any offset or count.

10       For all of the disk data caching performed through system memory 116, the FC 112a acts as a central authority for dynamically allocating, deallocating and keeping track of buffers. If there are two or more FCs 112, each has exclusive control over its own assigned portion of system memory 116. In all of these transactions, the requested buffers are locked  
15       during the period between the initial request and the release request. This prevents corruption of the data by other clients.

20       Also in the situation where there are two or more FCs, each file system on the disks is assigned to a particular one of the FCs! FC #0 runs a process called FC\_VICE\_PRESIDENT, which maintains a list of which file systems are assigned to which FC. When a client processor (for example an NC 110) is about to  
25       make an LNFS request designating a particular file system, it first sends the fsid in a message to the FC\_VICE\_PRESIDENT asking which FC controls the specified file system. The FC\_VICE\_PRESIDENT responds, and the client processor sends the LNFS request to the designated FC. The client processor  
30       also maintains its own list of fsid/FC pairs as it discovers them, so as to minimize the number of such requests to the FC\_VICE\_PRESIDENT.

#### STORAGE PROCESSOR HARDWARE ARCHITECTURE

35       In the file server 100, each of the storage processors 114 can interface the VME bus 120 with up

**SUBSTITUTE SHEET**

-43-

to 10 different SCSI buses. Additionally, it can do so at the full usage rate of an enhanced block transfer protocol of 55MB per second.

5 Fig. 5 is a block diagram of one of the SPs 114a. SP 114b is identical. SP 114a comprises a microprocessor 510, which may be a Motorola 68020 microprocessor operating at 20MHz. The microprocessor 510 is coupled over a 32-bit microprocessor data bus 512 with CPU memory 514, which may include up to 1MB  
10 of static RAM. The microprocessor 510 accesses instructions, data and status on its own private bus 512, with no contention from any other source. The microprocessor 510 is the only master of bus 512.

15 The low order 16 bits of the microprocessor data bus 512 interface with a control bus 516 via a bidirectional buffer 518. The low order 8 bits of the control bus 516 interface with a slow speed bus 520 via another bidirectional buffer 522. The slow speed bus 520 connects to an MFP 524, similar to the MFP 224  
20 in NC 110a (Fig. 3), and with a PROM 526, similar to PROM 220 on NC 110a. The PROM 526 comprises 128K bytes of EPROM which contains the functional code for SP 114a. Due to the width and speed of the EPROM 526, the functional code is copied to CPU memory 514 upon  
25 reset for faster execution.

30 MFP 524, like the MFP 224 on NC 110a, comprises a Motorola 68901 multifunction peripheral device. It provides the functions of a vectored interrupt controller, individually programmable I/O pins, four timers and a UART. The UART functions provide serial  
35 communications across an RS 232 bus (not shown in Fig. 5) for debug monitors and diagnostics. Two of the four timing functions may be used as general-purpose timers by the microprocessor 510, either independently or in cascaded fashion. A third timer function provides the refresh clock for a DMA controller

described below, and the fourth timer generates the  
 UART clock. Additional information on the MFP 524 can  
 be found in "MC 68901 Multi-Function Peripheral  
 Specification," by Motorola, Inc., which is  
 5 incorporated herein by reference. The eight  
 general-purpose I/O bits provided by MFP 524 are  
 configured according to the following table:

	<u>Bit</u>	<u>Direction</u>	<u>Definition</u>
10	7	input	Power Failure is Imminent - This functions as an early warning.
	6	input	SCSI Attention - A composite of the SCSI. Attentions from all 10 SCSI channels.
15	5	input	Channel Operation Done - A composite of the channel done bits from all 13 channels of the DMA controller, described below.
20	4	output	DMA Controller Enable. Enables the DMA Controller to run.
25	3	input	VMEbus Interrupt Done - Indicates the completion of a VMEbus Interrupt.
	2	input	Command Available - Indicates that the SP'S Command Fifo, described below, contains one or more command pointers.
30	1	output	External Interrupts Disable. Disables externally generated interrupts to the microprocessor 510.
35	0	output	Command Fifo Enable. Enables operation of the SP'S Command Fifo. Clears the Command Fifo when reset.

Commands are provided to the SP 114a from the VME  
 bus 120 via a bidirectional buffer 530, a local data  
 40 bus 532, and a command FIFO 534. The command FIFO 534  
 is similar to the command FIFOs 290 and 390 on NC 110a  
 and FC 112a, respectively, and has a depth of 256 32-  
 bit entries. The command FIFO 534 is a write-only  
 register as seen on the VME bus 120, and as a read-  
 45 only register as seen by microprocessor 510. If the

**SUBSTITUTE SHEET**

-45-

FIFO is full at the beginning of a write from the VME bus, a VME bus error is generated. Pointers are removed from the command FIFO 534 in the order received, and only by the microprocessor 510. Command available status is provided through I/O bit 4 of the MFP 524, and as long as one or more command pointers are still within the command FIFO 534, the command available status remains asserted.

As previously mentioned, the SP 114a supports up to 10 SCSI buses or channels 540a-540j. In the typical configuration, buses 540a-540i support up to 3 SCSI disk drives each, and channel 540j supports other SCSI peripherals such as tape drives, optical disks, and so on. Physically, the SP 114a connects to each of the SCSI buses with an ultra-miniature D sub connector and round shielded cables. Six 50-pin cables provide 300 conductors which carry 18 signals per bus and 12 grounds. The cables attach at the front panel of the SP 114a and to a commutator board at the disk drive array. Standard 50-pin cables connect each SCSI device to the commutator board. Termination resistors are installed on the SP 114a.

The SP 114a supports synchronous parallel data transfers up to 5MB per second on each of the SCSI buses 540, arbitration, and disconnect/reconnect services. Each SCSI bus 540 is connected to a respective SCSI adaptor 542, which in the present embodiment is an AIC 6250 controller IC manufactured by Adaptec Inc., Milpitas, California, operating in the non-multiplexed address bus mode. The AIC 6250 is described in detail in "AIC 6250 Functional Specification," by Adaptec Inc., which is incorporated herein by reference. The SCSI adaptors 542 each provide the necessary hardware interface and low-level electrical protocol to implement its respective SCSI channel.

**SUBSTITUTE SHEET**

-46-

5 The 8-bit data port of each of the SCSI adaptors 542 is connected to port A of a respective one of a set of ten parity FIFOs 544a-544j. The FIFOs 544 are the same as FIFOs 240, 260 and 270 on NC 110a, and are connected and configured to provide parity covered data transfers between the 8-bit data port of the respective SCSI adaptors 542 and a 36-bit (32-bit plus 4 bits of parity) common data bus 550. The FIFOs 544 provide handshake, status, word assembly/disassembly and speed matching FIFO buffering for this purpose. 10 The FIFOs 544 also generate and check parity for the 32-bit bus, and for RAID 5 implementations they accumulate and check redundant data and accumulate recovered data.

15 All of the SCSI adaptors 542 reside at a single location of the address space of the microprocessor 510, as do all of the parity FIFOs 544. The microprocessor 510 selects individual controllers and FIFOs for access in pairs, by first programming a pair select register (not shown) to point to the desired pair and then reading from or writing to the control register address of the desired chip in the pair. The microprocessor 510 communicates with the control registers on the SCSI adaptors 542 via the control bus 25 516 and an additional bidirectional buffer 546, and communicates with the control registers on FIFOs 544 via the control bus 516 and a bidirectional buffer 552. Both the SCSI adaptors 542 and FIFOs 544 employ 8-bit control registers, and register addressing of the FIFOs 544 is arranged such that such registers 30 alias in consecutive byte locations. This allows the microprocessor 510 to write to the registers as a single 32-bit register, thereby reducing instruction overhead.

35 The parity FIFOs 544 are each configured in their Adaptec 6250 mode. Referring to the Appendix, the

**SUBSTITUTE SHEET**



-47-

FIFOs 544 are programmed with the following bit settings in the Data Transfer Configuration Register:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
	0	WD Mode	(0)
5	1	Parity Chip	(1)
	2	Parity Correct Mode	(0)
	3	8/16 bits CPU & PortA interface	(0)
	4	Invert Port A address 0	(1)
	5	Invert Port A address 1	(1)
10	6	Checksum Carry Wrap	(0)
	7	Reset	(0)

The Data Transfer Control Register is programmed as follows:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
15	0	Enable PortA Req/Ack	(1)
	1	Enable PortB Req/Ack	(1)
	2	Data Transfer Direction	as desired
	3	CPU parity enable	(0)
20	4	PortA parity enable	(1)
	5	PortB parity enable	(1)
	6	Checksum Enable	(0)
	7	PortA Master	(0)

In addition, bit 4 of the RAM Access Control Register (Long Burst) is programmed for 8-byte bursts.

SCSI adaptors 542 each generate a respective interrupt signal, the status of which are provided to microprocessor 510 as 10 bits of a 16-bit SCSI interrupt register 556. The SCSI interrupt register 556 is connected to the control bus 516. Additionally, a composite SCSI interrupt is provided through the MFP 524 whenever any one of the SCSI adaptors 542 needs servicing.

An additional parity FIFO 554 is also provided in the SP 114a, for message passing. Again referring to the Appendix, the parity FIFO 554 is programmed with

**SUBSTITUTE SHEET**

the following bit settings in the Data Transfer Configuration Register:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
	0	WD Mode	(0)
5	1	Parity Chip	(1)
	2	Parity Correct Mode	(0)
	3	8/16 bits CPU & PortA interface	(1)
	4	Invert Port A address 0	(1)
	5	Invert Port A address 1	(1)
10	6	Checksum Carry Wrap	(0)
	7	Reset	(0)

The Data Transfer Control Register is programmed as follows:

	<u>Bit</u>	<u>Definition</u>	<u>Setting</u>
15	0	Enable PortA Req/Ack	(0)
	1	Enable PortB Req/Ack	(1)
	2	Data Transfer Direction	as desired
	3	CPU parity enable	(0)
	4	PortA parity enable	(0)
20	5	PortB parity enable	(1)
	6	Checksum Enable	(0)
	7	PortA Master	(0)

In addition, bit 4 of the RAM Access Control Register (Long Burst) is programmed for 8-byte bursts.

25 Port A of FIFO 554 is connected to the 16-bit control bus 516, and port B is connected to the common data bus 550. FIFO 554 provides one means by which the microprocessor 510 can communicate directly with the VME bus 120, as is described in more detail below.

30 The microprocessor 510 manages data movement using a set of 15 channels, each of which has an unique status which indicates its current state. Channels are implemented using a channel enable register 560 and a channel status register 562, both connected to  
 35 the control bus 516. The channel enable register 560

**SUBSTITUTE SHEET**

-49-

is a 16-bit write-only register, whereas the channel status register 562 is a 16-bit read-only register. The two registers reside at the same address to microprocessor 510. The microprocessor 510 enables a particular channel by setting its respective bit in channel enable register 560, and recognizes completion of the specified operation by testing for a "done" bit in the channel status register 562. The microprocessor 510 then resets the enable bit, which causes the respective "done" bit in the channel status register 562 to be cleared.

The channels are defined as follows:

CHANNEL FUNCTION

- 0:9 These channels control data movement to and from the respective FIFOs 544 via the communication data bus 550. When a FIFO is enabled and a request is received from it, the channel becomes ready. Once the channel has been serviced a status of done is generated.
- 11:10 These channels control data movement between a local data buffer 564, described below, and the VME bus 120. When enabled the channel becomes ready. Once the channel has been serviced a status of done is generated.
- 12 When enabled, this channel causes the DRAM in local data buffer 564 to be refreshed based on a clock which is generated by the MFP 524. The refresh consists of a burst of 16 rows. This channel does not generate a status of done.
- 13 The microprocessor's communication FIFO 554 is serviced by this channel. When enable is set and the FIFO 554 asserts a request then the channel becomes ready. This channel generates a status of done.
- 14 Low latency writes from microprocessor 510 onto the VME bus 120 are controlled by this channel. When this channel is enabled data is moved from a special 32 bit register, described below, onto the VME bus 120. This channel generates a done status.

**SUBSTITUTE SHEET**

-50-

15 This is a null channel for which neither a ready status nor done status is generated.

5 Channels are prioritized to allow servicing of the more critical requests first. Channel priority is assigned in a descending order starting at channel 14. That is, in the event that all channels are requesting service, channel 14 will be the first one served.

10 The common data bus 550 is coupled via a bidirectional register 570 to a 36-bit junction bus 572. A second bidirectional register 574 connects the junction bus 572 with the local data bus 532. Local data buffer 564, which comprises 1MB of DRAM, with parity, is coupled bidirectionally to the junction bus 15 572. It is organized to provide 256K 32-bit words with byte parity. The SP 114a operates the DRAMs in page mode to support a very high data rate, which requires bursting of data instead of random single-word accesses. It will be seen that the local data 20 buffer 564 is used to implement a RAID (redundant array of inexpensive disks) algorithm, and is not used for direct reading and writing between the VME bus 120 and a peripheral on one of the SCSI buses 540.

25 A read-only register 576, containing all zeros, is also connected to the junction bus 572. This register is used mostly for diagnostics, initialization, and clearing of large blocks of data in system memory 116.

30 The movement of data between the FIFOs 544 and 554, the local data buffer 564, and a remote entity such as the system memory 116 on the VME bus 120, is all controlled by a VME/FIFO DMA controller 580. The VME/FIFO DMA controller 580 is similar to the VME/FIFO DMA controller 272 on network controller 110a (Fig. 3), and is described in the Appendix. Briefly, it 35 includes a bit slice engine 582 and a dual-port static RAM 584. One port of the dual-port static RAM 584 communicates over the 32-bit microprocessor data bus

**SUBSTITUTE SHEET**

-51-

512 with microprocessor 510, and the other port communicates over a separate 16-bit bus with the bit slice engine 582. The microprocessor 510 places command parameters in the dual-port RAM 584, and uses the channel enables 560 to signal the VME/FIFO DMA controller 580 to proceed with the command. The VME/FIFO DMA controller is responsible for scanning the channel status and servicing requests, and returning ending status in the dual-port RAM 584. The dual-port RAM 584 is organized as 1K x 32 bits at the 32-bit port and as 2K x 16 bits at the 16-bit port. An example showing the method by which the microprocessor 510 controls the VME/FIFO DMA controller 580 is as follows. First, the microprocessor 510 writes into the dual-port RAM 584 the desired command and associated parameters for the desired channel. For example, the command might be, "copy a block of data from FIFO 544h out into a block of system memory 116 beginning at a specified VME address." Second, the microprocessor sets the channel enable bit in channel enable register 560 for the desired channel.

At the time the channel enable bit is set, the appropriate FIFO may not yet be ready to send data. Only when the VME/FIFO DMA controller 580 does receive a "ready" status from the channel, will the controller 580 execute the command. In the meantime, the DMA controller 580 is free to execute commands and move data to or from other channels.

When the DMA controller 580 does receive a status of "ready" from the specified channel, the controller fetches the channel command and parameters from the dual-ported RAM 584 and executes. When the command is complete, for example all the requested data has been copied, the DMA controller writes status back into the dual-port RAM 584 and asserts "done" for the channel in channel status register 562. The microprocessor

**SUBSTITUTE SHEET**

-52-

510 is then interrupted, at which time it reads channel status register 562 to determine which channel interrupted. The microprocessor 510 then clears the channel enable for the appropriate channel and checks the ending channel status in the dual-port RAM 584.

In this way a high-speed data transfer can take place under the control of DMA controller 580, fully in parallel with other activities being performed by microprocessor 510. The data transfer takes place over busses different from microprocessor data bus 512, thereby avoiding any interference with microprocessor instruction fetches.

The SP 114a also includes a high-speed register 590, which is coupled between the microprocessor data bus 512 and the local data bus 532. The high-speed register 590 is used to write a single 32-bit word to an VME bus target with a minimum of overhead. The register is write only as viewed from the microprocessor 510. In order to write a word onto the VME bus 120, the microprocessor 510 first writes the word into the register 590, and the desired VME target address into dual-port RAM 584. When the microprocessor 510 enables the appropriate channel in channel enable register 560, the DMA controller 580 transfers the data from the register 590 into the VME bus address specified in the dual-port RAM 584. The DMA controller 580 then writes the ending status to the dual-port RAM and sets the channel "done" bit in channel status register 562.

This procedure is very efficient for transfer of a single word of data, but becomes inefficient for large blocks of data. Transfers of greater than one word of data, typically for message passing, are usually performed using the FIFO 554.

The SP 114a also includes a series of registers 592, similar to the registers 282 on NC 110a (Fig. 3)

**SUBSTITUTE SHEET**

-53-

and the registers 382 on FC 112a (Fig. 4). The details of these registers are not important for an understanding of the present invention.

5        STORAGE PROCESSOR OPERATION

      The 30 SCSI disk drives supported by each of the SPs 114 are visible to a client processor, for example one of the file controllers 112, either as three large, logical disks or as 30 independent SCSI drives, depending on configuration. When the drives are visible as three logical disks, the SP uses RAID 5 design algorithms to distribute data for each logical drive on nine physical drives to minimize disk arm contention. The tenth drive is left as a spare. The RAID 5 algorithm (redundant array of inexpensive drives, revision 5) is described in "A Case For a Redundant Arrays of Inexpensive Disks (RAID)", by Patterson et al., published at ACM SIGMOD Conference, Chicago, Ill., June 1-3, 1988, incorporated herein by reference.

      In the RAID 5 design, disk data are divided into stripes. Data stripes are recorded sequentially on eight different disk drives. A ninth parity stripe, the exclusive-or of eight data stripes, is recorded on a ninth drive. If a stripe size is set to 8K bytes, a read of 8K of data involves only one drive. A write of 8K of data involves two drives: a data drive and a parity drive. Since a write requires the reading back of old data to generate a new parity stripe, writes are also referred to as modify writes. The SP 114a supports nine small reads to nine SCSI drives concurrently. When stripe size is set to 8K, a read of 64K of data starts all eight SCSI drives, with each drive reading one 8K stripe worth of data. The parallel operation is transparent to the caller client.

**SUBSTITUTE SHEET**

-54-

The parity stripes are rotated among the nine drives in order to avoid drive contention during write operations. The parity stripe is used to improve availability of data. When one drive is down, the SP 114a can reconstruct the missing data from a parity stripe. In such case, the SP 114a is running in error recovery mode. When a bad drive is repaired, the SP 114a can be instructed to restore data on the repaired drive while the system is on-line.

When the SP 114a is used to attach thirty independent SCSI drives, no parity stripe is created and the client addresses each drive directly.

The SP 114a processes multiple messages (transactions, commands) at one time, up to 200 messages per second. The SP 114a does not initiate any messages after initial system configuration. The following SP 114a operations are defined:

- 01 No Op
- 02 Send Configuration Data
- 03 Receive Configuration Data
- 05 Read and Write Sectors
- 06 Read and Write Cache Pages
- 07 IOCTL Operation
- 08 Dump SP 114a Local Data Buffer
- 09 Start/Stop A SCSI Drive
- 0C Inquiry
- 0E Read Message Log Buffer
- 0F Set SP 114a Interrupt

The above transactions are described in detail in the above-identified application entitled MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE. For and understanding of the invention, it will be useful to describe the function and operation of only two of these commands: read and write sectors, and read and write cache pages.

**SUBSTITUTE SHEET**



-55-

Read and Write Sectors

This command, issued usually by an FC 112, causes the SP 114a to transfer data between a specified block of system memory and a specified series of contiguous sectors on the SCSI disks. As previously described in connection with the file controller 112, the particular sectors are identified in physical terms. In particular, the particular disk sectors are identified by SCSI channel number (0-9), SCSI ID on that channel number (0-2), starting sector address on the specified drive, and a count of the number of sectors to read or write. The SCSI channel number is zero if the SP 114a is operating under RAID 5.

The SP 114a can execute up to 30 messages on the 30 SCSI drives simultaneously. Unlike most of the commands to an SP 114, which are processed by microprocessor 510 as soon as they appear on the command FIFO 534, read and write sectors commands (as well as read and write cache memory commands) are first sorted and queued. Hence, they are not served in the order of arrival.

When a disk access command arrives, the microprocessor 510 determines which disk drive is targeted and inserts the message in a queue for that disk drive sorted by the target sector address. The microprocessor 510 executes commands on all the queues simultaneously, in the order present in the queue for each disk drive. In order to minimize disk arm movements, the microprocessor 510 moves back and forth among queue entries in an elevator fashion.

If no error conditions are detected from the SCSI disk drives, the command is completed normally. When a data check error condition occurs and the SP 114a is configured for RAID 5, recovery actions using redundant data begin automatically. When a drive is down while the SP 114a is configured for RAID 5,

**SUBSTITUTE SHEET**

-56-

recovery actions similar to data check recovery take place.

#### Read/Write Cache Pages

5           This command is similar to read and write sectors, except that multiple VME addresses are provided for transferring disk data to and from system memory 116. Each VME address points to a cache page in system memory 116, the size of which is also specified in the  
10           command. When transferring data from a disk to system memory 116, data are scattered to different cache pages; when writing data to a disk, data are gathered from different cache pages in system memory 116. Hence, this operation is referred to as a scatter-  
15           gather function.

          The target sectors on the SCSI disks are specified in the command in physical terms, in the same manner that they are specified for the read and write sectors command. Termination of the command with or without  
20           error conditions is the same as for the read and write sectors command.

          The dual-port RAM 584 in the DMA controller 580 maintains a separate set of commands for each channel controlled by the bit slice engine 582. As each  
25           channel completes its previous operation, the microprocessor 510 writes a new DMA operation into the dual-port RAM 584 for that channel in order to satisfy the next operation on a disk elevator queue.

          The commands written to the DMA controller 580  
30           include an operation code and a code indicating whether the operation is to be performed in non-block mode, in standard VME block mode, or in enhanced block mode. The operation codes supported by DMA controller 580 are as follows:

**SUBSTITUTE SHEET**

	<u>OP CODE</u>	<u>OPERATION</u>	
	0	NO-OP	
5	1	ZEROES -> BUFFER	Move zeros from zeros register 576 to local data buffer 564.
10	2	ZEROES -> FIFO	Move zeros from zeros register 576 to the currently selected FIFO on common data bus 550.
15	3	ZEROES -> VMEbus	Move zeros from zeros register 576 out onto the VME bus 120. Used for initializing cache buffers in system memory 116.
20			
25	4	VMEbus -> BUFFER	Move data from the VME bus 120 to the local data buffer 564. This operation is used during a write, to move target data intended for a down drive into the buffer for participation in redundancy generation. Used only for RAID 5 application.
30			
35			
40	5	VMEbus -> FIFO	New data to be written from VME bus onto a drive. Since RAID 5 requires redundancy data to be generated from data that is buffered in local data buffer 564, this operation will be used only if the SP 114a is not configured for RAID 5.
45			
50	6	VMEbus -> BUFFER & FIFO	Target data is moved from VME bus 120 to a SCSI

**SUBSTITUTE SHEET**

-58-

5 device and is also captured in the local data buffer 564 for participation in redundancy generation. Used only if SP 114a is configured for RAID 5 operation.

10 7 BUFFER -> VMEbus This operation is not used.

15 8 BUFFER -> FIFO Participating data is transferred to create redundant data or recovered data on a disk drive. Used only in RAID 5 applications.

20 9 FIFO -> VMEbus This operation is used to move target data directly from a disk drive onto the VME bus 120.

25 A FIFO -> BUFFER Used to move participating data for recovery and modify operations. Used only in RAID 5 applications.

30 B FIFO -> VMEbus & BUFFER This operation is used to save target data for participation in data recovery. Used only in RAID 5 applications.

35

SYSTEM MEMORY

40 Fig. 6 provides a simplified block diagram of the preferred architecture of one of the system memory cards 116a. Each of the other system memory cards are the same. Each memory card 116 operates as a slave on the enhanced VME bus 120 and therefore requires no on-board CPU. Rather, a timing control block 610 is sufficient to provide the necessary slave control operations. In particular, the timing control block

45

**SUBSTITUTE SHEET**

-59-

610, in response to control signals from the control portion of the enhanced VME bus 120, enables a 32-bit wide buffer 612 for an appropriate direction transfer of 32-bit data between the enhanced VME bus 120 and a multiplexer unit 614. The multiplexer 614 provides a multiplexing and demultiplexing function, depending on data transfer direction, for a six megabit by seventy-two bit word memory array 620. An error correction code (ECC) generation and testing unit 622 is also connected to the multiplexer 614 to generate or verify, again depending on transfer direction, eight bits of ECC data. The status of ECC verification is provided back to the timing control block 610.

#### 15 ENHANCED VME BUS PROTOCOL

VME bus 120 is physically the same as an ordinary VME bus, but each of the NCs and SPs include additional circuitry and firmware for transmitting data using an enhanced VME block transfer protocol. The enhanced protocol is described in detail in the above-identified application entitled ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER, and summarized in the Appendix hereto. Typically transfers of LNFS file data between NCs and system memory, or between SPs and system memory, and transfers of packets being routed from one NC to another through system memory, are the only types of transfers that use the enhanced protocol in server 100. All other data transfers on VME bus 120 use either conventional VME block transfer protocols or ordinary non-block transfer protocols.

#### MESSAGE PASSING

As is evident from the above description, the different processors in the server 100 communicate with each other via certain types of messages. In

**SUBSTITUTE SHEET**

-60-

software, these messages are all handled by the messaging kernel, described in detail in the MULTIPLE FACILITY OPERATING SYSTEM ARCHITECTURE application cited above. In hardware, they are implemented as follows.

Each of the NCs 110, each of the FCs 112, and each of the SPs 114 includes a command or communication FIFO such as 290 on NC 110a. The host 118 also includes a command FIFO, but since the host is an unmodified purchased processor board, the FIFO is emulated in software. The write port of the command FIFO in each of the processors is directly addressable from any of the other processors over VME bus 120.

Similarly, each of the processors except SPs 114 also includes shared memory such as CPU memory 214 on NC 110a. This shared memory is also directly addressable by any of the other processors in the server 100.

If one processor, for example network controller 110a, is to send a message or command to a second processor, for example file controller 112a, then it does so as follows. First, it forms the message in its own shared memory (e.g., in CPU memory 214 on NC 110a). Second, the microprocessor in the sending processor directly writes a message descriptor into the command FIFO in the receiving processor. For a command being sent from network controller 110a to file controller 112a, the microprocessor 210 would perform the write via buffer 284 on NC 110a, VME bus 120, and buffer 384 on file controller 112a.

The command descriptor is a single 32-bit word containing in its high order 30 bits a VME address indicating the start of a quad-aligned message in the sender's shared memory. The low order two bits indicate the message type as follows:

**SUBSTITUTE SHEET**

-61-

	<u>Type</u>	<u>Description</u>
	0	Pointer to a new message being sent
	1	Pointer to a reply message
	2	Pointer to message to be forwarded
5	3	Pointer to message to be freed; also message acknowledgment

All messages are 128-bytes long.

When the receiving processor reaches the command descriptor on its command FIFO, it directly accesses the sender's shared memory and copies it into the receiver's own local memory. For a command issued from network controller 110a to file controller 112a, this would be an ordinary VME block or non-block mode transfer from NC CPU memory 214, via buffer 284, VME bus 120 and buffer 384, into FC CPU memory 314. The FC microprocessor 310 directly accesses NC CPU memory 214 for this purpose over the VME bus 120.

When the receiving processor has received the command and has completed its work, it sends a reply message back to the sending processor. The reply message may be no more than the original command message unaltered, or it may be a modified version of that message or a completely new message. If the reply message is not identical to the original command message, then the receiving processor directly accesses the original sender's shared memory to modify the original command message or overwrite it completely. For replies from the FC 112a to the NC 110a, this involves an ordinary VME block or non-block mode transfer from the FC 112a, via buffer 384, VME bus 120, buffer 284 and into NC CPU memory 214. Again, the FC microprocessor 310 directly accesses NC CPU memory 214 for this purpose over the VME bus 120.

Whether or not the original command message has been changed, the receiving processor then writes a reply message descriptor directly into the original sender's command FIFO. The reply message descriptor

**SUBSTITUTE SHEET**

-62-

contains the same VME address as the original command message descriptor, and the low order two bits of the word are modified to indicate that this is a reply message. For replies from the FC 112a to the NC 110a, the message descriptor write is accomplished by microprocessor 310 directly accessing command FIFO 290 via buffer 384, VME bus 120 and buffer 280 on the NC. Once this is done, the receiving processor can free the buffer in its local memory containing the copy of the command message.

When the original sending processor reaches the reply message descriptor on its command FIFO, it wakes up the process that originally sent the message and permits it to continue. After examining the reply message, the original sending processor can free the original command message buffer in its own local shared memory.

As mentioned above, network controller 110a uses the buffer 284 data path in order to write message descriptors onto the VME bus 120, and uses VME/FIFO DMA controller 272 together with parity FIFO 270 in order to copy messages from the VME bus 120 into CPU memory 214. Other processors read from CPU memory 214 using the buffer 284 data path.

File controller 112a writes message descriptors onto the VME bus 120 using the buffer 384 data path, and copies messages from other processors' shared memory via the same data path. Both take place under the control of microprocessor 310. Other processors copy messages from CPU memory 314 also via the buffer 384 data path.

Storage processor 114a writes message descriptors onto the VME bus using high-speed register 590 in the manner described above, and copies messages from other processors using DMA controller 580 and FIFO 554. The SP 114a has no shared memory, however, so it uses a

**SUBSTITUTE SHEET**



-63-

buffer in system memory 116 to emulate that function. That is, before it writes a message descriptor into another processor's command FIFO, the SP 114a first copies the message into its own previously allocated  
5 buffer in system memory 116 using DMA controller 580 and FIFO 554. The VME address included in the message descriptor then reflects the VME address of the message in system memory 116.

10 In the host 118, the command FIFO and shared memory are both emulated in software.

The invention has been described with respect to particular embodiments thereof, and it will be understood that numerous modifications and variations are possible within the scope of the invention.

**SUBSTITUTE SHEET**

-64-

APPENDIX AVME/FIFO DMA Controller

5 In storage processor 114a, DMA controller 580  
manages the data path under the direction of the  
microprocessor 510. The DMA controller 580 is a  
microcoded 16-bit bit-slice implementation executing  
pipelined instructions at a rate of one each 62.5ns.  
10 It is responsible for scanning the channel status 562  
and servicing request with parameters stored in the  
dual-ported ram 584 by the microprocessor 510. Ending  
status is returned in the ram 584 and interrupts are  
generated for the microprocessor 510.

15 Control Store. The control store contains the  
microcoded instructions which control the DMA  
controller 580. The control store consists of 6 1K x  
8 proms configured to yield a 1K x 48 bit microword.  
Locations within the control store are addressed by  
the sequencer and data is presented at the input of  
20 the pipeline registers.

Sequencer. The sequencer controls program flow by  
generating control store addresses based upon pipeline  
data and various status bits. The control store  
address consists of 10 bits. Bits 8:0 of the control  
25 store address derive from a multiplexer having as its  
inputs either an ALU output or the output of an  
incrementer. The incrementer can be preloaded with  
pipeline register bits 8:0, or it can be incremented  
as a result of a test condition. The 1K address range  
30 is divided into two pages by a latched flag such that  
the microprogram can execute from either page.  
Branches, however remain within the selected page.  
Conditional sequencing is performed by having the test  
condition increment the pipeline provided address. A  
35 false condition allows execution from the pipeline  
address while a true condition causes execution from

**SUBSTITUTE SHEET**

-65-

the address + 1. The alu output is selected as an address source in order to directly vector to a routine or in order to return to a calling routine. Note that when calling a subroutine the calling routine must reside within the same page as the subroutine or the wrong page will be selected on the return.

ALU. The alu comprises a single IDT49C402A integrated circuit. It is 16 bits in width and most closely resembles four 2901s with 64 registers. The alu is used primarily for incrementing, decrementing, addition and bit manipulation. All necessary control signals originate in the control store. The IDT HIGH PERFORMANCE CMOS 1988 DATA BOOK, incorporated by reference herein, contains additional information about the alu.

Microword. The 48 bit microword comprises several fields which control various functions of the DMA controller 580. The format of the microword is defined below along with mnemonics and a description of each function.

25	AI<8:0> 47:39	(Alu Instruction bits 8:0) The AI bits provide the instruction for the 49C402A alu. Refer to the IDT data book for a complete definition of the alu instructions. Note that the I9 signal input of the 49C402A is always low.
30	CIN            38	(Carry INput) This bit forces the carry input to the alu.
35	RA<5:0> 37:32	(Register A address bits 5:0) These bits select one of 64 registers as the "A" operand for the alu. These bits also provide literal bits 15:10 for the alu bus.
40	RB<5:0> 31:26	(Register B address bits 5:0) These bits select one of 64 registers as the "B" operand for the alu. These bits also provide literal bits 9:4 for the alu bus.

**SUBSTITUTE SHEET**

5 LFD 25 (Latched Flag Data) When set this bit causes the selected latched flag to be set. When reset this bit causes the selected latched flag to be cleared. This bits also functions as literal bit 3 for the alu bus.

10 LFS<2:0> 24:22 (Latched Flag Select bits 2:0) The meaning of these bits is dependent upon the selected source for the alu bus. In the event that the literal field is selected as the bus source then LFS<2:0> function as literal bits <2:0> otherwise the bits are used to select one of the latched flags.

15

LFS<2:0>    SELECTED FLAG

20                    0        This value selects a null flag.

25                    1        When set this bit enables the buffer clock. When reset this bit disables the buffer clock.

30                    2        When this bit is cleared VME bus transfers, buffer operations and RAS are all disabled.

35                    3        NOT USED

                      4        When set this bit enables VME bus transfers.

40                    5        When set this bit enables buffer operations.

45                    6        When set this bit asserts the row address strobe to the dram buffer.

                      7        When set this bit selects page 0 of the control store.

50 SRC<1,0> 20,21 (alu bus SourCe select bits 1,0) These bits select the data source to be enabled onto the alu bus.

**SUBSTITUTE SHEET**

-67-

SRC<1:0> Selected Source

- 0 alu
- 1 dual ported ram
- 2 literal
- 3 reserved-not defined

PF<2:0> 19:17 (Pulsed Flag select bits 2:0) These bits select a flag/signal to be pulsed.

PF<2:0> Flag

- 0 null
- 1 SGL\_CLK  
generates a single transition of buffer clock.
- 2 SET\_VB  
forces vme and buffer enable to be set.
- 3 CL\_PERR  
clears buffer parity error status.
- 4 SET\_DN  
set channel done status for the currently selected channel.
- 5 INC\_ADR  
increment dual ported ram address.
- 6:7 RESERVED - NOT DEFINED

DEST<3:0> 16:13 (DESTINATION select bits 3:0) These bits select one of 10 destinations to be loaded from the alu bus.

DEST<3:0> Destination

- 0 null
- 1 WR\_RAM  
causes the data on the alu bus to be written to the dual ported ram.  
D<15:0> -> ram<15:0>
- 2 WR\_BADD

**SUBSTITUTE SHEET**

-68-

loads the data from the alu bus  
into the dram address counters.

5	3	<p>D&lt;14:7&gt; -&gt; mux addr&lt;8:0&gt; WR_VADL loads the data from the alu bus into the least significant 2 bytes of the VME address register. D&lt;15:2&gt; -&gt; VME addr&lt;15:2&gt; D1 -&gt; ENB_tional registers D&lt;15:2&gt; -&gt; VME addr&lt;15:2&gt; D1 -&gt; ENB_ENH D0 -&gt; ENB_BLK</p>
10		
15	4	<p>WR_VADH loads the most significant 2 bytes of the VME address register. D&lt;15:0&gt; -&gt; VME addr&lt;31:16&gt;</p>
20		
25	5	<p>WR_RADD loads the dual ported ram address counters. D&lt;10:0&gt; -&gt; ram addr &lt;10:0&gt;</p>
30	6	<p>WR_WCNT loads the word counters. D15 -&gt; count enable* D&lt;14:8&gt; -&gt; count &lt;6:0&gt;</p>
35	7	<p>WR_CO loads the co-channel select register. D&lt;7:4&gt; -&gt; CO&lt;3:0&gt;</p>
40	8	<p>WR_NXT loads the next-channel select register. D&lt;3:0&gt; -&gt; NEXT&lt;3:0&gt;</p>
45	9	<p>WR_CUR loads the current-channel select register. D&lt;3:0&gt; -&gt; CURR &lt;3:0&gt;</p>
50	10:14	RESERVED - NOT DEFINED
	15	<p>JUMP causes the control store sequencer to select the alu data bus. D&lt;8:0&gt; -&gt; CS_A&lt;8:0&gt;</p>

**SUBSTITUTE SHEET**

TEST<3:0> 12:9 (TEST condition select bits 3:0)  
 Select one of 16 inputs to the test  
 multiplexor to be used as the carry  
 input to the incrementer.

5

TEST<3:0> Condition

10

0	FALSE	-always false
1	TRUE	-always true
2	ALU_COUT	-carry output of alu
3	ALU_EQ	-equals output of alu

15

4	ALU_OVR	-alu overflow
5	ALU_NEG	-alu negative

20

6	XFR_DONE	-transfer complete
7	PAR_ERR	-buffer parity error
8	TIMOUT	-bus operation timeout

25

9	ANY_ERR	-any error status
---	---------	-------------------

14:10	RESERVED	-NOT DEFINED
-------	----------	--------------

30

15	CH_RDY	-next channel ready
----	--------	---------------------

NEXT\_A<8:0> 8:0 (NEXT Address bits 8:0) Selects an  
 instructions from the current page of the  
 control store for execution.

35

Dual Ported Ram. The dual ported ram is the  
 medium by which command, parameters and status are  
 communicated between the DMA controller 580 and the  
 microprocessor 510. The ram is organized as 1K x 32 at  
 the master port and as 2K x 16 at the DMA port. The  
 ram may be both written and read at either port.

40

The ram is addressed by the DMA controller 580 by  
 loading an 11 bit address into the address counters.  
 Data is then read into bidirectional registers and the  
 address counter is incremented to allow read of the  
 next location.

45

**SUBSTITUTE SHEET**

Writing the ram is accomplished by loading data from the processor into the registers after loading the ram address. Successive writes may be performed on every other processor cycle.

5 The ram contains current block pointers, ending status, high speed bus address and parameter blocks. The following is the format of the ram:

OFFSET	31	0
10 0	CURR POINTER 0	STATUS 0
	-----	
4	INITIAL POINTER 0	
	-----	
15	-----	
58	CURR POINTER B	STATUS B
	-----	
5C	INITIAL POINTER B	
	-----	
20 60	not used	not used
	-----	
64	not used	not used
	-----	
25 68	CURR POINTER D	STATUS D
	-----	
6C	INITIAL POINTER D	
	-----	
70	not used	STATUS E
	-----	
30 74	HIGH SPEED BUS ADDRESS 31:2 0 0	
	-----	
78	PARAMETER BLOCK 0	
	-----	
35	-----	
??	PARAMETER BLOCK n	
	-----	

40 The Initial Pointer is a 32 bit value which points the first command block of a chain. The current pointer is a sixteen bit value used by the DMA controller 580 to point to the current command block. The current command block pointer should be  
 45 initialized to 0x0000 by the microprocessor 510 before enabling the channel. Upon detecting a value of 0x0000

**SUBSTITUTE SHEET**



in the current block pointer the DMA controller 580 will copy the lower 16 bits from the initial pointer to the current pointer. Once the DMA controller 580 has completed the specified operations for the parameter block the current pointer will be updated to point to the next block. In the event that no further parameter blocks are available the pointer will be set to 0x0000.

The status byte indicates the ending status for the last channel operation performed. The following status bytes are defined:

<u>STATUS MEANING</u>	
0	NO ERRORS
1	ILLEGAL OP CODE
2	BUS OPERATION TIMEOUT
3	BUS OPERATION ERROR
4	DATA PATH PARITY ERROR

The format of the parameter block is:

OFFSET	31	0
0	FORWARD LINK	
4	NOT USED	WORD COUNT
8	VME ADDRESS 31:2, ENH, BLK	
C	TERM 0	OP 0   BUF ADDR 0
	.	
	.	
	.	
C+(4Xn)	TERM n	OP n   BUF ADDR n

FORWARD LINK - The forward link points to the first word of the next parameter block for execution. It allows several parameter blocks to be initialized and chained to create a sequence of operations for execution. The forward pointer has the following format:

**SUBSTITUTE SHEET**

-72-

A31:A2,0,0

The format dictates that the parameter block must start on a quad byte boundary. A pointer of 0x00000000 is a special case which indicates no forward link exists.

WORD COUNT - The word count specifies the number of quad byte words that are to be transferred to or from each buffer address or to/from the VME address. A word count of 64K words may be specified by initializing the word count with the value of 0. The word count has the following format:

[D15|D14|D13|D12|D11|D10|D9|D8|D7|D6|D5|D4|D3|D2|D1|D0|

The word count is updated by the DMA controller 580 at the completion of a transfer to/from the last specified buffer address. Word count is not updated after transferring to/from each buffer address and is therefore not an accurate indicator of the total data moved to/from the buffer. Word count represents the amount of data transferred to the VME bus or one of the FIFOs 544 or 554.

VME ADDRESS - The VME address specifies the starting address for data transfers. Thirty bits allows the address to start at any quad byte boundary.

ENH - This bit when set selects the enhanced block transfer protocol described in the above-cited ENHANCED VMEBUS PROTOCOL UTILIZING PSEUDOSYNCHRONOUS HANDSHAKING AND BLOCK MODE DATA TRANSFER application, to be used during the VME bus transfer. Enhanced protocol will be disabled automatically when performing any transfer to or from 24 bit or 16 bit address space, when the starting address is not 8 byte aligned or when the word count is not even.

BLK - This bit when set selects the conventional VME block mode protocol to be used during the VME bus transfer. Block mode will be disabled automatically

**SUBSTITUTE SHEET**

-73-

when performing any transfer to or from 16 bit address space.

5        BUF ADDR - The buffer address specifies the starting buffer address for the adjacent operation. Only 16 bits are available for a 1M byte buffer and as a result the starting address always falls on a 16 byte boundary. The programmer must ensure that the starting address is on a modulo 128 byte boundary. The buffer address is updated by the DMA controller 580 after completion of each data burst.

10        |A19|A18|A17|A16|A15|A14|A13|A12|A11|A10|A9|A8|A7|A6|A5|A4|

15        TERM - The last buffer address and operation within a parameter block is identified by the terminal bit. The DMA controller 580 continues to fetch buffer addresses and operations to perform until this bit is encountered. Once the last operation within the parameter block is executed the word counter is updated and if not equal to zero the series of operations is repeated. Once the word counter reaches zero the forward link pointer is used to access the next parameter block.

20        |0|0|0|0|0|0|0|0|0|0|T|

25        OP - Operations are specified by the op code. The op code byte has the following format:

      |0|0|0|0|OP3|OP2|OP1|OP0|

The op codes are listed below ("FIFO" refers to any of the FIFOs 544 or 554):

**SUBSTITUTE SHEET**

	<u>OP CODE</u>	<u>OPERATION</u>
	0	NO-OP
	1	ZEROES -> BUFFER
	2	ZEROES -> FIFO
5	3	ZEROES -> VMEbus
	4	VMEbus -> BUFFER
	5	VMEbus -> FIFO
	6	VMEbus -> BUFFER & FIFO
	7	BUFFER -> VMEbus
10	8	BUFFER -> FIFO
	9	FIFO -> VMEbus
	A	FIFO -> BUFFER
	B	FIFO -> VMEbus & BUFFER
	C	RESERVED
15	D	RESERVED
	E	RESERVED
	F	RESERVED

SUBSTITUTE SHEET

-75-

APPENDIX BEnhanced VME Block Transfer Protocol

5 The enhanced VME block transfer protocol is a  
VMEbus compatible pseudo-synchronous fast transfer  
handshake protocol for use on a VME backplane bus  
having a master functional module and a slave  
functional module logically interconnected by a data  
transfer bus. The data transfer bus includes a data  
strobe signal line and a data transfer acknowledge  
10 signal line. To accomplish the handshake, the master  
transmits a data strobe signal of a given duration on  
the data strobe line. The master then awaits the  
reception of a data transfer acknowledge signal from  
the slave module on the data transfer acknowledge  
15 signal line. The slave then responds by transmitting  
data transfer acknowledge signal of a given duration  
on the data transfer acknowledge signal line.

Consistent with the pseudo-synchronous nature of  
the handshake protocol, the data to be transferred is  
20 referenced to only one signal depending upon whether  
the transfer operation is a READ or WRITE operation.

In transferring data from the master functional  
unit to the slave, the master broadcasts the data to  
be transferred. The master asserts a data strobe  
25 signal and the slave, in response to the data strobe  
signal, captures the data broadcast by the master.  
Similarly, in transferring data from the slave to the  
master, the slave broadcasts the data to be  
transferred to the master unit. The slave then  
30 asserts a data transfer acknowledge signal and the  
master, in response to the data transfer acknowledge  
signal, captures the data broadcast by the slave.

The fast transfer protocol, while not essential to  
the present invention, facilitates the rapid transfer  
35 of large amounts of data across a VME backplane bus by  
substantially increasing the data transfer rate.

**SUBSTITUTE SHEET**

-76-

These data rates are achieved by using a handshake wherein the data strobe and data transfer acknowledge signals are functionally decoupled and by specifying high current drivers for all data and control lines.

5       The enhanced pseudo-synchronous method of data transfer (hereinafter referred to as "fast transfer mode") is implemented so as to comply and be compatible with the IEEE VME backplane bus standard. The protocol utilizes user-defined address modifiers,  
10       defined in the VMEbus standard, to indicate use of the fast transfer mode. Conventional VMEbus functional units, capable only of implementing standard VMEbus protocols, will ignore transfers made using the fast transfer mode and, as a result, are fully compatible  
15       with functional units capable of implementing the fast transfer mode.

      The fast transfer mode reduces the number of bus propagations required to accomplish a handshake from  
20       four propagations, as required under conventional VMEbus protocols, to only two bus propagations. Likewise, the number of bus propagations required to effect a BLOCK READ or BLOCK WRITE data transfer is reduced. Consequently, by reducing the propagations  
25       across the VMEbus to accomplish handshaking and data transfer functions, the transfer rate is materially increased.

      The enhanced protocol is described in detail in the above-cited ENHANCED VMEBUS PROTOCOL application, and will only be summarized here. Familiarity with the  
30       conventional VME bus standards is assumed.

      In the fast transfer mode handshake protocol, only two bus propagations are used to accomplish a handshake, rather than four as required by the conventional protocol. At the initiation of a data  
35       transfer cycle, the master will assert and deassert DS0\* in the form of a pulse of a given duration. The

**SUBSTITUTE SHEET**

-77-

deassertion of DS0\* is accomplished without regard as to whether a response has been received from the slave. The master then waits for an acknowledgement from the slave. Subsequent pulsing of DS0\* cannot occur until a responsive DTACK\* signal is received from the slave. Upon receiving the slave's assertion of DTACK\*, the master can then immediately reassert data strobe, if so desired. The fast transfer mode protocol does not require the master to wait for the deassertion of DTACK\* by the slave as a condition precedent to subsequent assertions of DS0\*. In the fast transfer mode, only the leading edge (i.e., the assertion) of a signal is significant. Thus, the deassertion of either DS0\* or DTACK\* is completely irrelevant for completion of a handshake. The fast transfer protocol does not employ the DS1\* line for data strobe purposes at all.

The fast transfer mode protocol may be characterized as pseudo-synchronous as it includes both synchronous and asynchronous aspects. The fast transfer mode protocol is synchronous in character due to the fact that DS0\* is asserted and deasserted without regard to a response from the slave. The asynchronous aspect of the fast transfer mode protocol is attributable to the fact that the master may not subsequently assert DS0\* until a response to the prior strobe is received from the slave. Consequently, because the protocol includes both synchronous and asynchronous components, it is most accurately classified as "pseudo-synchronous."

The transfer of data during a BLOCK WRITE cycle in the fast transfer protocol is referenced only to DS0\*. The master first broadcasts valid data to the slave, and then asserts DS0 to the slave. The slave is given a predetermined period of time after the assertion of DS0\* in which to capture the data. Hence, slave

**SUBSTITUTE SHEET**

-78-

modules must be prepared to capture data at any time, as DTACK\* is not referenced during the transfer cycle.

Similarly, the transfer of data during a BLOCK READ cycle in the fast transfer protocol is referenced only to DTACK\*. The master first asserts DS0\*. The slave then broadcasts data to the master and then asserts DTACK\*. The master is given a predetermined period of time after the assertion of DTACK in which to capture the data. Hence, master modules must be prepared to capture data at any time as DS0 is not referenced during the transfer cycle.

Fig. 7, parts A through C, is a flowchart illustrating the operations involved in accomplishing the fast transfer protocol BLOCK WRITE cycle. To initiate a BLOCK WRITE cycle, the master broadcasts the memory address of the data to be transferred and the address modifier across the DTB bus. The master also drives interrupt acknowledge signal (IACK\*) high and the LWORD\* signal low 701. A special address modifier, for example "1F," broadcast by the master indicates to the slave module that the fast transfer protocol will be used to accomplish the BLOCK WRITE.

The starting memory address of the data to be transferred should reside on a 64-bit boundary and the size of block of data to be transferred should be a multiple of 64 bits. In order to remain in compliance with the VMEbus standard, the block must not cross a 256 byte boundary without performing a new address cycle.

The slave modules connected to the DTB receive the address and the address modifier broadcast by the master across the bus and receive LWORD\* low and IACK\* high 703. Shortly after broadcasting the address and address modifier 701, the master drives the AS\* signal low 705. The slave modules receive the AS\* low signal 707. Each slave individually determines whether it

**SUBSTITUTE SHEET**



-79-

will participate in the data transfer by determining whether the broadcasted address is valid for the slave in question 709. If the address is not valid, the data transfer does not involve that particular slave and it ignores the remainder of the data transfer cycle.

5 The master drives WRITE\* low to indicate that the transfer cycle about to occur is a WRITE operation 711. The slave receives the WRITE\* low signal 713 and, knowing that the data transfer operation is a WRITE operation, awaits receipt of a high to low transition on the DS0\* signal line 715. The master will wait until both DTACK\* and BERR\* are high 718, which indicates that the previous slave is no longer driving the DTB.

15 The master proceeds to place the first segment of the data to be transferred on data lines D00 through D31, 719. After placing data on D00 through D31, the master drives DS0\* low 721 and, after a predetermined interval, drives DS0\* high 723.

20 In response to the transition of DS0\* from high to low, respectively 721 and 723, the slave latches the data being transmitted by the master over data lines D00 through D31, 725. The master places the next segment of the data to be transferred on data lines D00 through D31, 727, and awaits receipt of a DTACK\* signal in the form of a high to low transition signal, 729 in Fig. 7B.

25 Referring to Fig. 7B, the slave then drives DTACK\* low, 731, and, after a predetermined period of time, drives DTACK high, 733. The data latched by the slave, 725, is written to a device, which has been selected to store the data 735. The slave also increments the device address 735. The slave then waits for another transition of DS0\* from high to low 737.

**SUBSTITUTE SHEET**