US 20070299991A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2007/0299991 A1

Giebel (43) Pub. Date: Dec. 27, 2007

(54) DMA MODULE AND OPERATING SYSTEM THEREFOR

(76) Inventor: **Burkhard Giebel**, Denzlingen (DE)

Correspondence Address:
O'SHEA, GETZ & KOSAKOWSKI, P.C.
1500 MAIN ST.
SUITE 912
SPRINGFIELD, MA 01115 (US)

(21) Appl. No.: **11/485,750**

(22) Filed: **Jul. 13, 2006**

Related U.S. Application Data

(62) Division of application No. 10/751,668, filed on Jan. 5, 2004, now Pat. No. 7,293,120.

(30) **Foreign Application Priority Data**

Jan. 3, 2003 (DE)..... 103 00 030.5

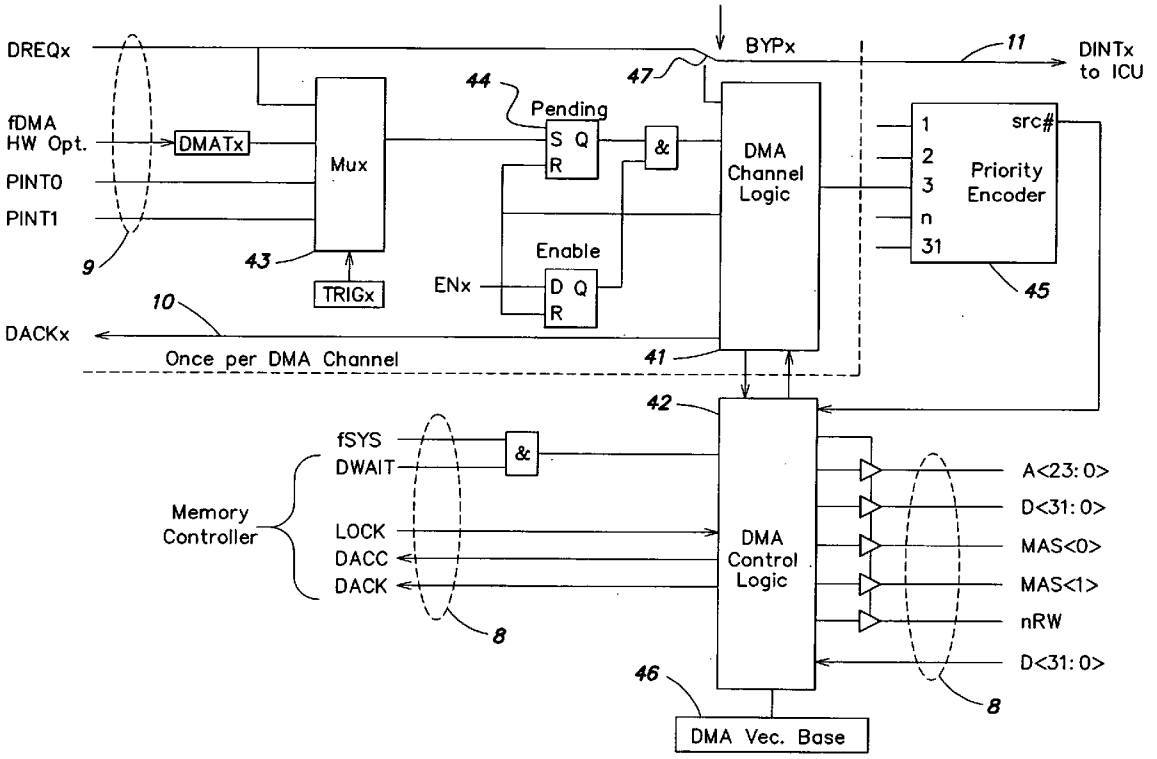
Publication Classification

(51) **Int. Cl.**
G06F 13/28 (2006.01)

(52) **U.S. Cl.** **710/22**

(57) **ABSTRACT**

A DMA module is described which, in order to read or write a memory location within a DMA process, accesses by a reading operation a memory location of an addressable memory (**5**) identified by a first address (**46**) in order to read there at least one second address (**52**); which advances the second address (**52**) to an adjacent memory location, and implements a write access or read access at a memory location identified by the second address (**52**); and which finally stores the second address at the memory location identified by the first address (**46**).



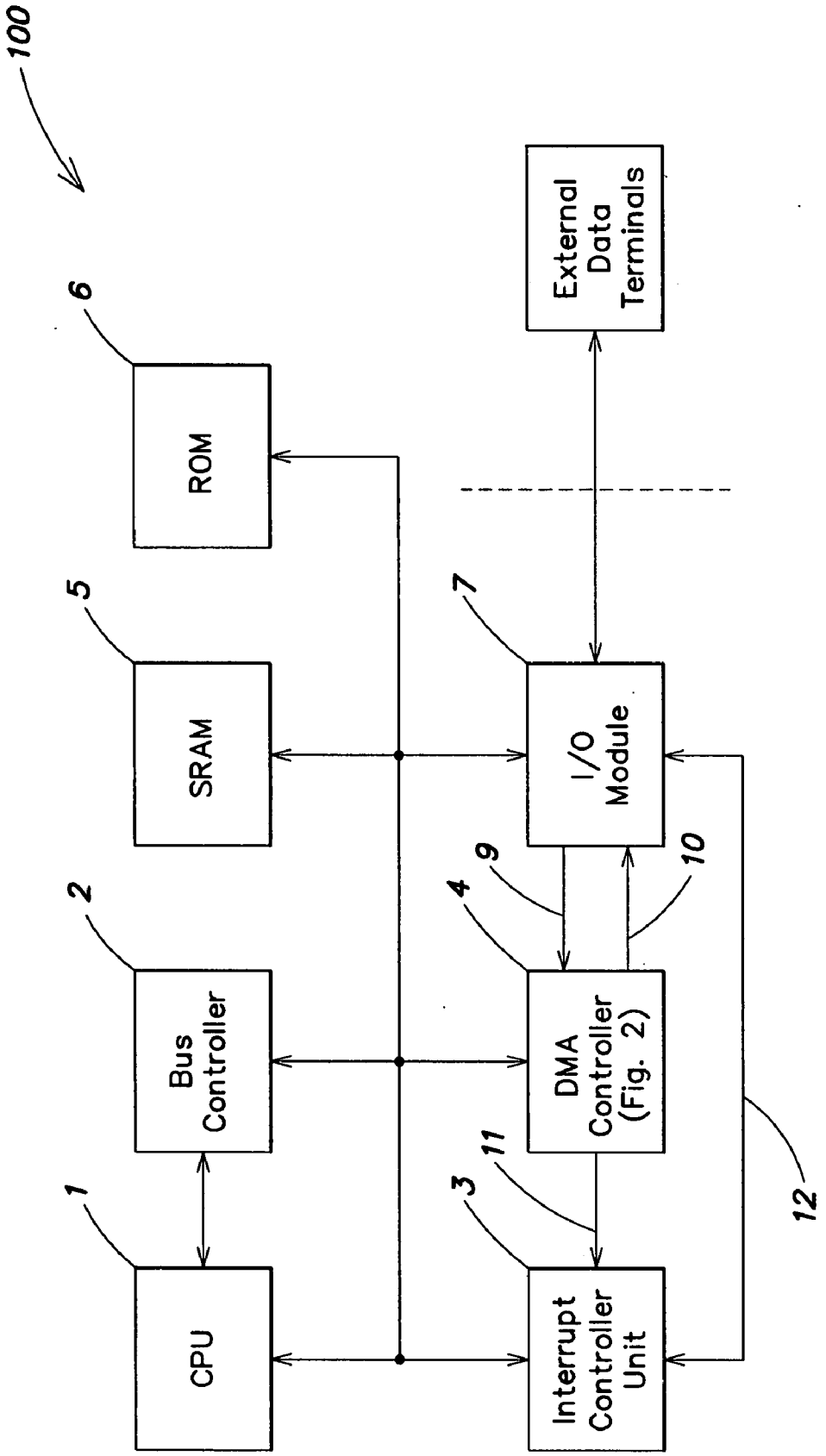

100
External Data Terminals
6
ROM
5
SRAM
7
I/O Module
2
Bus Controller
9
10
4
DMA Controller (Fig. 2)
11
1
CPU
3
Interrupt Controller Unit
12

FIG. 1

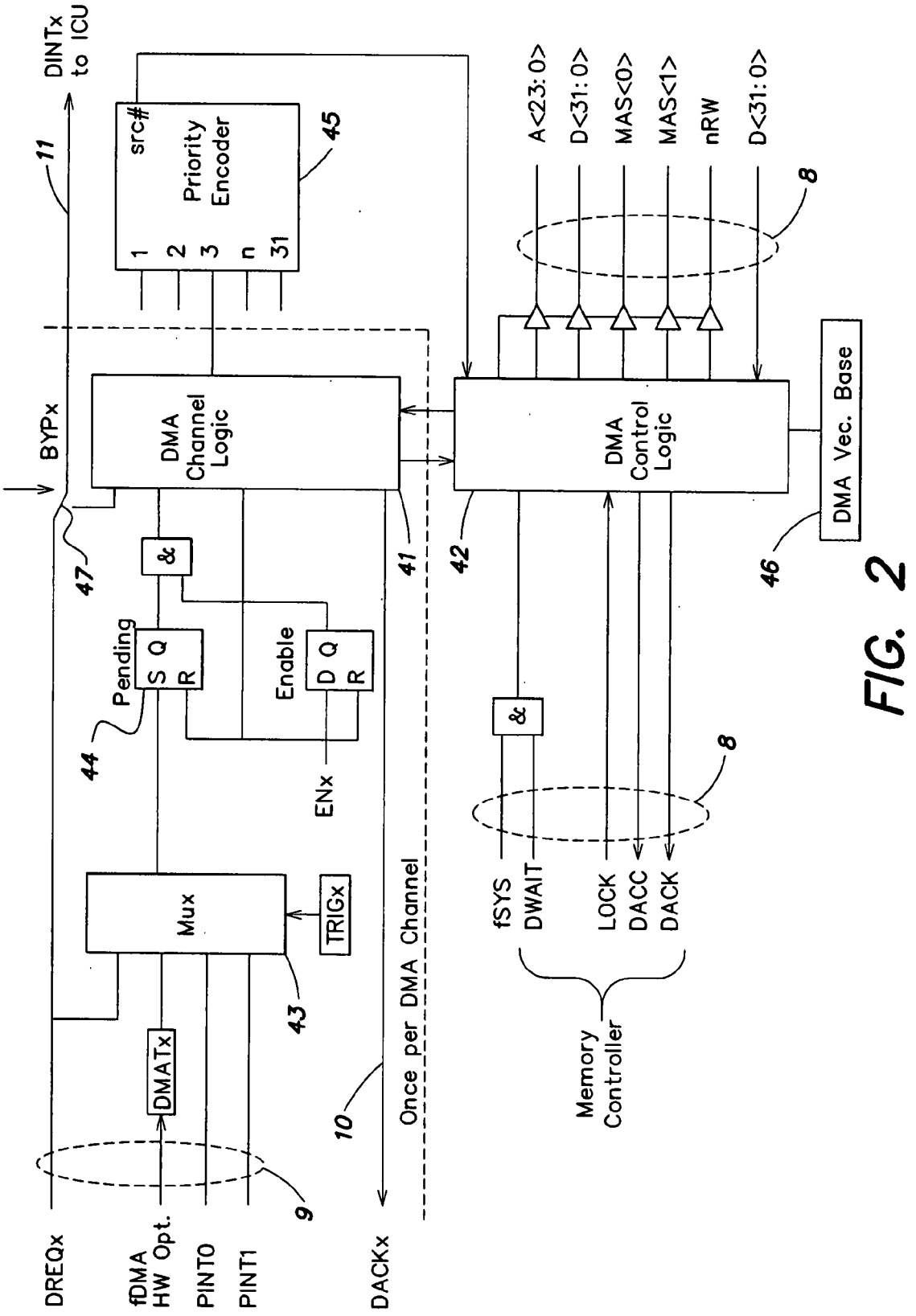

DINTx to ICU
11
src#
Priority Encoder
45
1
2
3
n
31
A<23:0>
D<31:0>
MAS<0>
MAS<1>
nRW
D<31:0>
8
BYPx
DMA Channel Logic
DMA Control Logic
DMA Vec. Base
47
&
41
42
46
Pending
S Q
R
44
Enable
D Q
R
ENx
&
8
Mux
TRIGx
43
DMATx
9
10
Once per DMA Channel
fSYS
DWAIT
LOCK
DACC
DACK
Memory Controller
DREQx
fDMA HW Opt.
PINT0
PINT1
DACKx

FIG. 2

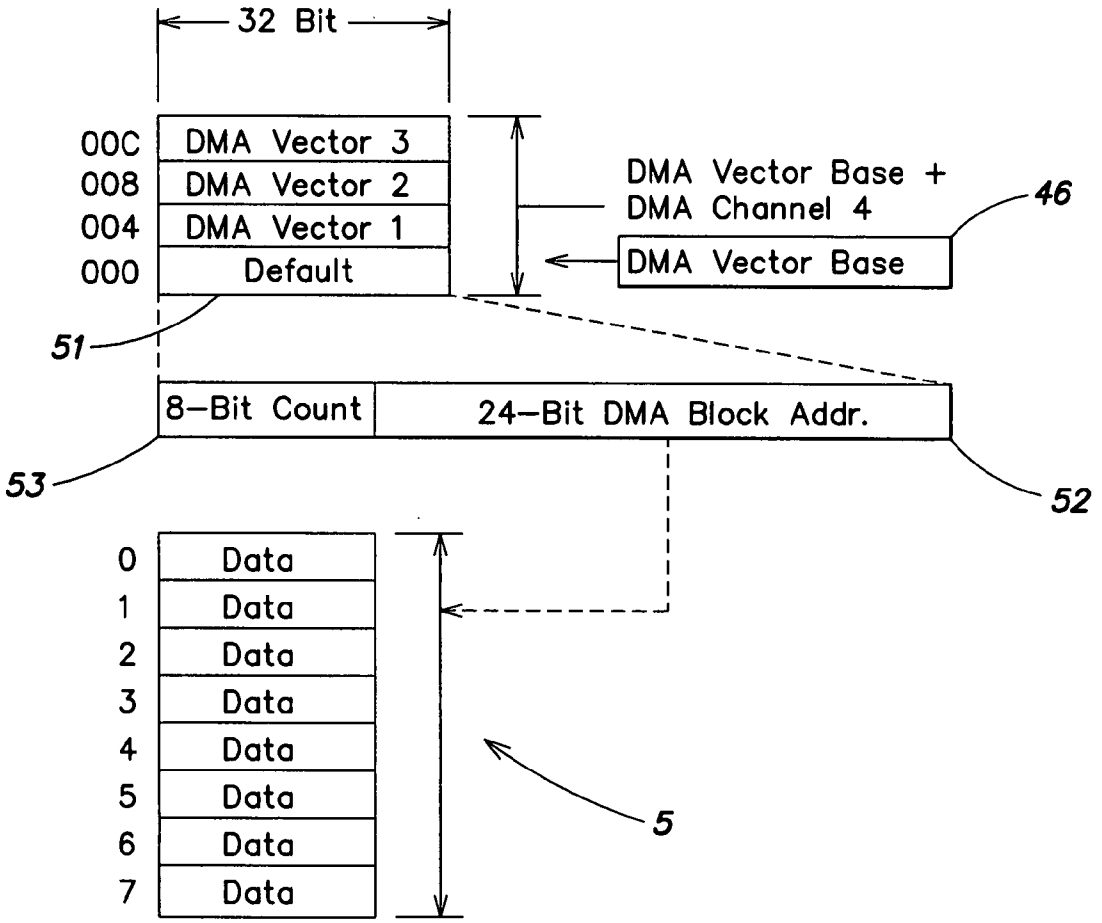

32 Bit
00C
008
004
000
DMA Vector 3
DMA Vector 2
DMA Vector 1
Default
DMA Vector Base +
DMA Channel 4
DMA Vector Base
46
51
8-Bit Count
24-Bit DMA Block Addr.
53
52
0
1
2
3
4
5
6
7
Data
5

FIG. 3

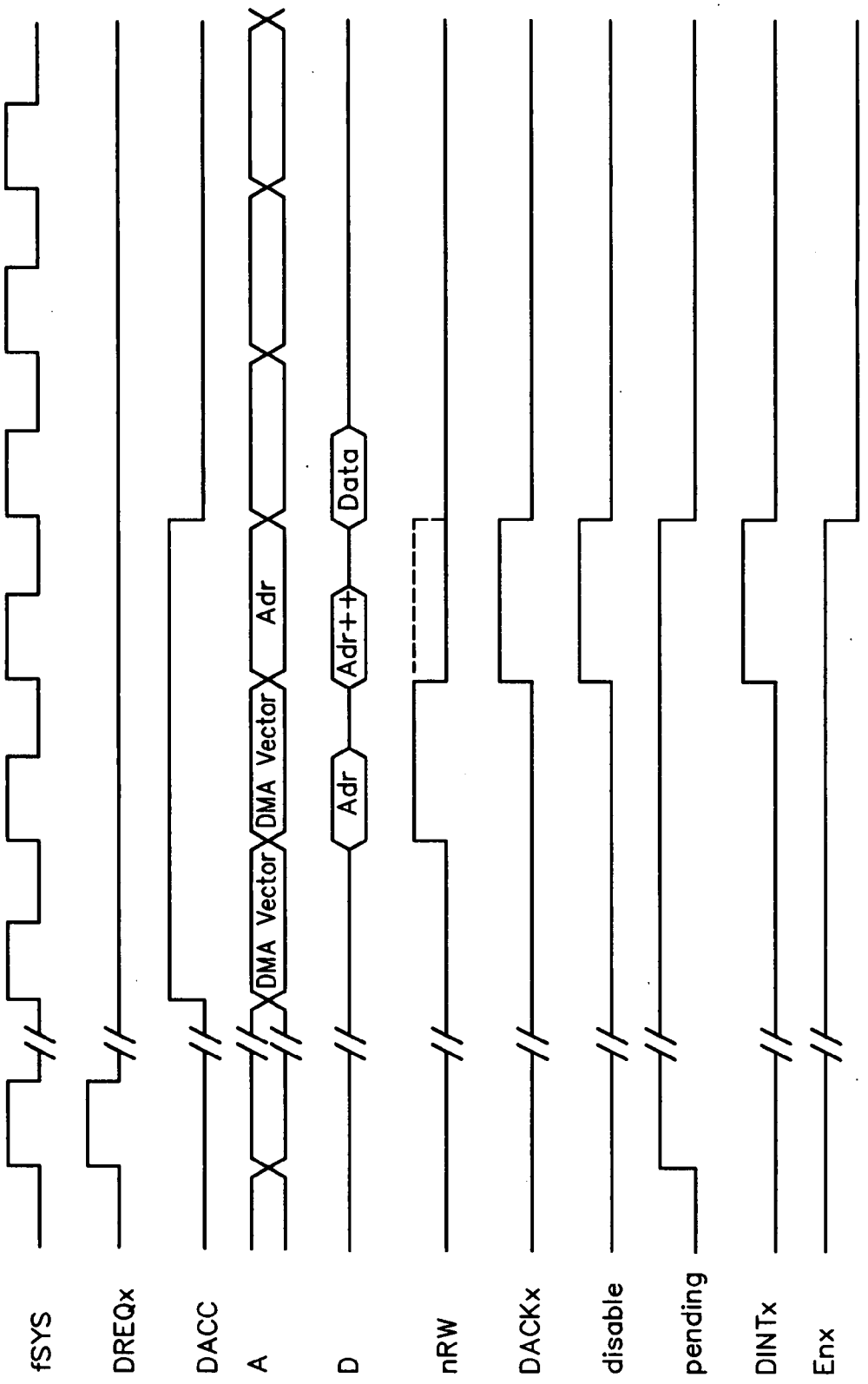

fSYS
DREQx
DACC
A
D
nRW
DACKx
disable
pending
DINTx
Enx
DMA Vector
DMA Vector
Adr
Adr
Adr++
Data

FIG. 4

DMA MODULE AND OPERATING SYSTEM THEREFOR

PRIORITY INFORMATION

[0001] This application is a divisional of co-pending Ser. No. 10/751,668 filed Jan. 5, 2004.

BACKGROUND OF THE INVENTION

[0002] The invention relates to the field of microcontrollers, and in particular to the field of direct memory access (DMA).

[0003] DMA modules are employed in microcomputer controller systems and microcontroller systems to relieve the CPU from routinely recurring data transfer tasks. A DMA module may be viewed as a kind of specialized processor which receives a specification for a memory area which must be accessed, for example, a start address and stop address, or a start address and a number of subsequent memory locations, and which outputs the addresses of a specified range in rapid succession to an address bus in order to enable rapid reading from or writing to the memory area. A DMA module thus enables the rapid transfer of data between memory areas, or between one memory area and a peripheral device, without taking up any of the processing power of the CPU. A DMA module thus relieves the CPU from simple data transfer tasks, thereby increasing the average achievable performance of the CPU.

[0004] If the CPU and the DMA module share a common bus for access to the memory, then for every bus cycle in which the DMA module accesses the memory for a read or write operation the CPU must be halted to prevent the CPU from attempting to initiate access at the same time. Although during this procedure, known as "cycle stealing" the operation of the CPU is slightly retarded, the number of CPU cycles lost is significantly less than if the CPU itself had to control the data transfer. Hence, use of the DMA module enhances the performance of the CPU.

[0005] A DMA module always proceeds from a start address to a stop address. In the conventional approach, the two addresses are stored in internal registers of the DMA module. As a result, it is possible with a single bus cycle to output an address, to read or write a memory location identified by the address, and to advance an address counter for the following read or write operation.

[0006] A DMA module with two registers of this type is able to perform only one DMA process at a time. If a running DMA process has to be interrupted to perform a DMA process of higher priority, the register contents of the currently running, lower priority process with its current values from the DMA module must be swapped out and temporarily stored to allow the relevant parameters of the higher priority process to be loaded and executed. This swapping out and loading of addresses slows down the DMA module, and the microcontroller along with it, and is therefore not desirable.

[0007] A conceivable approach would be to equip a DMA module of this type with a plurality of register pairs that would hold the parameters of the different DMA processes. As a result, to interrupt a currently running DMA process in favor of a higher-priority process the system would switch to a different register pair.

[0008] The problem with this approach, however, is the fact that registers of this type have a high space requirement on a semiconductor substrate, this requirement being five times the requirement of a memory element of identical size within RAM. This large space requirement makes fabrication of multi-process DMA modules expensive.

[0009] Therefore, there is a need for DMA module which has a small space requirement, and a method of operating a DMA module in which such small-area DMA modules may be employed.

SUMMARY OF THE INVENTION

[0010] A DMA module includes a conventional address generator to perform a write or read access to a memory location of an addressable memory, and an address counter to advance a stored address to an adjacent memory location. The address counter can not act on an internal register of the DMA module but instead be configured so that between reading an address value from the memory and writing the address value to the memory, this counter is advanced once. As a result, the memory location of the memory at which the address value is read or written takes on the function of a register conventionally integrated in the DMA module. This approach reduces the space requirement of the DMA module, and yields the additional advantage that the DMA module may be employed to control a large number of DMA processes that may mutually interrupt each other, simply by providing a plurality of memory locations of the memory to store specifications of the DMA blocks.

[0011] Operation of the DMA module comprises the following steps:

[0012] a) Implementing a read access to a memory location identified by the first address so as to read there at least a second address which points to the next memory location that must be accessed in the current DMA process;

[0013] b) Advancing the second address to an adjacent memory location, storing the second address at the memory location identified by the first address, and

[0014] c) Implementing a write access or read access at a memory location identified by the second address (although it is unimportant whether or not the second address is first advanced and the memory location identified by it then accessed, or visa versa).

[0015] Since the parameters required to control the DMA process are present in the addressable memory, it is no longer necessary to first swap out register values in order to switch from one DMA process to another. As a result, rapid switching is therefore possible between different DMA processes.

[0016] In order to have the DMA module determine when a DMA process has been completed, it is appropriate before write accessing and read accessing the memory location identified by the second address to read a count value that is representative for the number of memory access operations yet to be performed, to decrement the count value, and then to re-store it after implementation of the write/read access to the second address as set forth in step c). In this way, the DMA module can determine from the count value when a DMA process has been completed, and terminate it at the right time.

[0017] To reduce the number of memory access operations, it is useful to record this count value during each read access of step a) and to log the value during the write access of step b). This procedure is implemented as long as the total bit number of the second address and of the count value do not exceed the width of data words read in a single data access, in other words, the width of the data bus. In this way, one bus cycle is sufficient to read or store the required parameters of one DMA process.

[0018] Another aspect of the invention involves the utilization of the normally frequently available interrupt request line of the peripheral module to transfer the DMA request signal by which a peripheral module "orders" a DMA transfer. To this end, the DMA module is advantageously looped into the request line to the interrupt controller, and is thus, in a manner of speaking, connected in series before the interrupt controller.

[0019] As a result, a design is created that is able to initiate a number of "cheap" (in terms of CPU performance) DMA transfers before an expensive interrupt is initiated. A design of this type is particularly suitable for collecting data to process it at the end of the process in a concentrated way with CPU participation but without detectable CPU overloading.

[0020] In an aspect of the invention, a data table may be transferred under DMA control between the peripheral module and RAM, but nevertheless initiate an CPU interrupt routine at the end of the process. In the interrupt routine, the CPU is able to process, for example, the current data table in a time-concentrated manner, then configure the DMA module to transfer the next data table and start this DMA sequence.

[0021] These and other objects, features and advantages of the present invention will become more apparent in light of the following detailed description of preferred embodiments thereof, as illustrated in the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWING

[0022] FIG. **1** is a block diagram illustration of a microcontroller system;

[0023] FIG. **2** is a block diagram of a DMA controller;

[0024] FIG. **3** illustrates a design of DMA vector table; and

[0025] FIG. **4** is a timing diagram of various signals occurring in the DMA controller of FIG. **2**.

DETAILED DESCRIPTION OF THE INVENTION

[0026] FIG. **1** is a block diagram illustration of a microcontroller system **100**. The microcontroller system **100** comprises a CPU **1**, a bus controller **2**, an interrupt manager unit (interrupt controller ICU) **3**, a DMA controller **4**, a static read-write memory (SRAM) **5**, a read-only memory (ROM) **6**, and an input/output module **7**. The components **1-7** are interconnected by an internal bus **8** with address lines, data lines and control lines. In addition, the I/O module **7** is connected to one or more external data terminals from which the module receives data to store in internal bus **8**, or to which terminals it outputs data from internal bus **8**. The I/O module **7**, DMA controller **8**, and ICU **3** are directly interconnected by a plurality of control lines **9-12** isolated from the bus **8**. The design and function of the control lines **9-12** is discussed in more detail in connection with FIG. **2**.

[0027] FIG. **2** shows in greater detail the design of DMA controller **4**. The functions of DMA controller **4** may be subdivided in to two areas, the first being the management of different processes that may initiate a DMA access, in this case various data sources or data sinks connected through I/O module **7** to the microcontroller system, i.e., deciding for which of the various processes a DMA will be implemented at a given point in time. A second part of the DMA controller **4** is responsible for controlling the actual memory access. The management circuits of these two areas are designated as DMA channel logic **41** or DMA control logic **42**. One DMA channel logic **41** is provided for each DMA channel, i.e., for each process that may be triggered for a DMA. An input multiplexer **43**, to which the signals identifying the requesting process x, namely DREQx, x=1, 2, 3, 3, etc. from the control line **9** are applied, sets a flag in one RS flip-flop **44** indicating the pending state of a DMA process. This flag is automatically reset by DMA channel logic **41** if the corresponding DMA cycle has terminated. An enabling flag EN masks the pending state flag from priority encoder **45**. The priority encoder **45** is connected to outputs from all DMA the channel logic circuits **41** and sends to its output an ordinal number corresponding to the channel logic circuit **41** with a pending DMA process has the highest priority level. The priority levels for the individual processes are recorded in the priority encoder **45**. In addition, the DMA channel logic **41** sends an acknowledgment signal DACKx through the control line **10** back to the triggering process, which signal indicates implementation of a DMA access and communicates to the process that—depending on the access direction—a data value read from the memory **5** is located on the internal bus **8** and may be accepted, or that a data value supplied from the process has been accepted and a new value must be provided.

[0028] A switch **47** is located in request signal line DREQX, x=1, 2, 3, which allows a signal arriving in line DREQX to be optionally fed through to the ICU **3**, where it is able to trigger an interrupt, or to supply a control signal generated by the DMA channel logic **41** to the ICU **3**. The function of the signal on the line **12** is to directly transfer data between the ICU **3** and the interface **7** in a situation in which request signal DREQx is fed through at the switch **47** to the signal on the line **11**.

[0029] Within one DMA cycle, the DMA control logic **42** controls the sequence of each individual access. In order to be able to manage the different DMA processes that may run in a time-overlapping fashion, the DMA control logic **42** accesses a DMA vector table **51** that is located in the SRAM **5**. A start address for this vector table **51** is recorded in a table initial register **46** of DMA control logic **42**. The entries of the DMA vector table may be 32-bit words that form a specification for a DMA process and each may be composed, as shown in FIG. **3**, of a 24-bit-wide address **52** that is the initial address of a block in the SRAM **5** in which a DMA access is to occur and of an 8-bit-wide count value that indicates the length of the block in bytes.

[0030] Each time before a DMA process begins, the address and count value are each written to the DMA vector table **51**, for example, by the CPU **1**. Given the width of

eight (8) bits for the count value assumed here in the example, a DMA block may have a maximum length of 2^8=256 bytes. It is of course evident that any other types of apportionment for DMA vectors between the initial address and the count value, and thus other block lengths, are conceivable.

[0031] Whenever the DMA control logic **42** of the priority encoder **45** receives the number of a DMA process for which an access is to be implemented, the control logic **42** calculates an access address by adding the fourfold multiple of the number supplied by the priority encoder **45** to the address of the table initial register **46**. If the number supplied by the priority encoder **45** is able to assume the values 1, 2 or 3, then the control logic **42** is able to access the fields of the vector table **51** identified in FIG. **3** as "DMA Vector 1", "DMA Vector 2", and "DMA Vector 3".

[0032] It is readily evident that the DMA controller **4** is easily adaptable in order to manage varying numbers of DMA processes without significant modifications to its circuit design. The limiting factor for the number of processes is simply the output width of the priority encoder **45**, i.e., the bit number of the line through which the encoder transmits the number of a DMA-triggering process to the control logic **42**, and the size of the DMA vector table **51**.

[0033] FIG. **4** illustrates the behavior of different signals within the microcontroller system over the course of a DMA process. Here f_{sys} designates the system clock at which both the CPU and the DMA controller **4** operate. A period of f_{sys} corresponds to one CPU clock cycle. A pulse of the signal DREQx (x=1, 2, 3, etc.) indicates a DMA request by the process x. Several cycles may elapse before the request is fulfilled. When this happens, signal DACC supplied by the DMA controller **4** to the memory control changes from low to high level in order to indicate that the DMA controller **4** has control of the bus and that the CPU **1** may not access it with either a write or a read operation. In a first clock cycle with DACC at high level, the DMA controller **4** uses the identity of the requesting process and the address recorded in the table initial register **46** to pass the address of one of vectors "DMA Vector 1", "DMA Vector 2" or "DMA Vector 3" to address lines A of the bus **8**. The SRAM **5**, in which the table S**1** of vectors is located, then sets the specification for a DMA block, recorded at the corresponding address and including of initial address and byte number to be sent, to the data lines of the bus **8**. These values are transferred by the DMA controller **4** into an address register or a counter.

[0034] In the next clock cycle, the address of the DMA vector is again set to the address lines of the bus **8**. At the same time, the initial address incremented by one and the count value decremented by one are passed to the data lines of the bus **8** where the previous DMA vector is overwritten by these values. In the following clock cycle, the incremented address is output to the address lines in order to—depending on the access direction of the DMA process—implement a write or read access at the corresponding memory location.

[0035] Whenever multiple write access and read access operations of a single DMA process follow in immediate succession, i.e., without interruption by a DMA process of higher priority, the step of reading the DMA vector may be eliminated for the second, and all subsequent, memory access operations since the specification stored there has already been stored in the registers of the DMA module. The memory locations identified by the DMA vectors thus have the function of registers that point to an address immediately before that memory address in the SRAM **5** at which the next write/read access is to take place. In other words, no access occurs at the actual address written to a DMA vector upon initialization of a DMA process but only at the subsequent addresses. Once the count value zero (0) is reached, the DMA process is terminated.

[0036] Of course, an alternative approach might be one in which, upon initialization of a DMA process at the memory location to which the assigned DMA vector must point, the size of the memory block to be processed and the first address are recorded at which a write or read operation should be implemented. In this case, the sequence of steps described above in reference to FIG. **4** would be slightly modified. Immediately after reading the memory location identified by the DMA vector, the write/read access to the address indicated there could follow, then an incremented address and a decremented count value would be stored at a location identified by the DMA vector in a subsequent clock cycle.

[0037] In either case, the DMA process requires three clock cycles in which the CPU **1** is halted to read or write a first memory location, and at least two clock cycles for each of the following memory locations. Although this operation is slower than in a DMA in which the addresses of the memory block to be processed are stored directly in registers of a DMA controller and do not have to be obtained from a swap-out memory, it is nevertheless significantly faster than if the CPU **1** itself has to perform these tasks. Since the number of registers required in the DMA controller is independent of the number of DMA processes that the DMA controller **4** is able to process simultaneously, the invention makes it very easy to design a DMA controller which is able to handle any desirable large number of DMA processes simultaneously.

[0038] Although the present invention has been shown and described with respect to several preferred embodiments thereof, various changes, omissions and additions to the form and detail thereof, may be made therein, without departing from the spirit and scope of the invention.

What is claimed is:

1. A method of operating a DMA module, comprising the steps:

a) implementing a read access to a memory location of an addressable memory (**5**) identified by a first address in order to read there at least one second address (**52**);

b) in any order: advancing the second address (**52**) to an adjacent memory location and storing the second address (**52**) at the memory location identified by the first address;

c) implementing a write or read access at a memory location identified by the second address (**52**).

2. The method of claim 1, characterized in that before each implementation of step b) a count value (**53**) is read, that the count value (**53**) is decremented or incremented and re-stored after implementation of step b), and that the method halts when the count value has reached a predetermined final value.

3. The method of claim 2, characterized in that the count value (**53**) is recorded during each read access of step a), and logged during each write access of step b).

4. The method of claim 1, characterized in that the write access or read access takes place within one cycle of a bus (**8**) through which the DMA module (**4**) is connected to the memory (**5**).

5. The method of claim 4, characterized in that the DMA module (**4**) halts a microcontroller (**1**) also connected to the memory (**5**) within a bus cycle in which the module accesses the memory (**5**).

6. The method of claim 1, including the step in which a DMA access is requested by a peripheral module, and the first address is selected based on the identity of the peripheral module.

7. The method of claim 6, characterized in that the peripheral module requests the DMA access through its interrupt request line.

8. The method of claim 6, characterized in that the interrupt request is suppressed in response to the triggering of the DMA access through the interrupt request line.

* * * * *