BUFFERING DATA THAT FLOWS BETWEEN BUSES OPERATING AT DIFFERENT FREQUENCIES

AUSGLEICHEN VON DATEN DIE ZWISCHEN VERSCHIEDENENLEITERN FLIESEN DIE AUF UNTERSCHIEDLICHEN FREQUENZEN OPERIEREN

MISE EN MEMOIRE TAMPON DE DONNEES S’ECOULANT ENTRE DES BUS FONCTIONNANT A DES FREQUENCES DIFFERENTES

Designated Contracting States:
CH DE FR GB IT LI SE

Priority: 27.10.1997 US 957856

Date of publication of application: 06.09.2000 Bulletin 2000/36

Proprietary: Emulex Design & Manufacturing Corporation
Costa Mesa, California 92626 (US)

Inventors:
• SU, Sam
Rowland Heights, CA 91748 (US)
• CHAU, Vi
Laguna Niguel, CA 92677 (US)
• TARR, Dan
Irvine, CA 92614 (US)

Representative: Vossius & Partner
Siebertstrasse 4
81675 München (DE)

References cited:
EP-A- 0 545 575

• JOHNSON W J ET AL: "DYNAMIC FIFO RAM MANAGER FOR A BUFFER INSERTION RING" IBM TECHNICAL DISCLOSURE BULLETIN, IBM CORP. NEW YORK, US, vol. 27, no. 4B, 1 September 1984 (1984-09-01), pages 2730-2731, XP000715192 ISSN: 0018-8689

• "ALGORITHM FOR MANAGING MULTIPLE FIRST-IN, FIRST-OUT QUEUES FROM A SINGLE SHARED RANDOM-ACCESS MEMORY" IBM TECHNICAL DISCLOSURE BULLETIN, IBM CORP. NEW YORK, US, vol. 32, no. 3B, 1 August 1989 (1989-08-01), pages 488-492, XP000029766 ISSN: 0018-8689

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).
DESCRIPTION

TECHNICAL FIELD

[0001] The invention relates to data buffering, and more particularly to buffering data that flows between buses operating at different frequencies.

BACKGROUND INFORMATION

[0002] A typical computer system includes multiple data buses that facilitate the flow of data between various components of the system. In general, the data buses are of different types that operate according to different functional standards, such as the Industry-Standard Architecture (ISA), Extended Industry-Standard Architecture (EISA), Peripheral Component Interface (PCI), and Small Computer System Interface (SCSI) standards. Among the characteristics that distinguish the different types of data buses is operating frequency, or the rate at which a bus transfers data among components residing on the bus. In a typical computer system, data routinely is transferred between devices located on different buses operating at different frequencies. For example, a microprocessor located on a bus operating at 50 MHz often must deliver data to a device located on a PCI bus operating at 8 MHz. To compensate for such disparities in operating frequency, the computer system must include a buffering system between the data buses.

[0003] Conventional buffering systems employ standard memory devices as first-in, first-out (FIFO) buffers. A typical FIFO buffer usually has two ports, one of which receives data from a first bus at a first data rate while the other delivers the data to a second bus at a second data rate. FIFO buffering schemes require careful synchronization of read and write controllers to prevent data from overflowing or underflowing the FIFO buffers. Some FIFO buffering schemes use multiple memory devices to create multiple FIFO buffers, such as in the well-known "double-buffering" techniques, where one of a pair of separately addressable buffers has data written to it while the other buffer is being read. The roles of the separate buffers are periodically switched.

[0004] Various techniques have been used to work around the buffer synchronization problem. One technique is to have the write side and the read side of a single buffer operate in a mutually exclusive fashion. Write control logic puts data into the buffer until the buffer is full, and then signals read control logic to start reading the content. When the reading is completed, the read control logic signals the write control logic to start filling the buffer again. This process continues until all data is transferred. This is a fool-proof method to guarantee the data integrity, but the drawback is long latency since at least one side of the buffer is idle at any given time.

[0005] Another method of determining FIFO full/empty status is by comparing the read and the write pointers of a FIFO. If the two pointers line up and the last operation is a write command, the FIFO is full. If the last operation is a read command when the two pointers lines up, the FIFO is empty. This method works well when the read and write control logic are both running at the same clock speed. When the two sides are running at two different clock speeds, there is a potential to produce an erroneous or inconclusive empty/full status, and thus there may be a danger of data corruption.

[0006] US-A-5 363 485 discloses a bus interface for connecting two busses. It includes a first and second bus interface circuitry, central buffer and control circuitry. The control circuitry includes two control sequence circuitries for tracking and controlling the channels of data within a FIFO device. The sequence control circuitry includes a circular queue for providing a predetermined number of slots, each slot capable of containing the identity and the status of the channel of data already resident in the FIFO device, and capable of containing the identity of the channel of data pending residence in the FIFO device. The sequence control circuitry further includes two pointers for traversing the circular queue, slot by slot.

[0007] The inventors have concluded that a better approach is needed to buffering data that flows between buses operating at different frequencies. The present invention provides a system and method for achieving this objective.

SUMMARY

[0008] The invention features a "virtual FIFO" system for use in buffering data between transacting buses that transfer data at different rates. The system includes a memory device partitioned into multiple banks, each of which is configured to operate as a distinct data buffer. Such partitioning may be done on a dynamic, "on-the-fly" basis. The invention eliminates the possibility of FIFO status ambiguity while maintaining FIFO efficiency by allowing both read and write operations to operate concurrently.

[0009] In the preferred embodiment, the memory device is a two-port (one read, one write) random access (RAM) device configured as a FIFO. A two-port RAM allows independent and random access to any location of the RAM from both the read and the write side of the FIFO. A write controller generates a write strobe to the RAM and controls write address generation. A read controller is responsible for controlling read address generation.

[0010] While the RAM FIFO allows read and write access concurrently, for any particular bank, only one of the two operations is allowed, such that the read controller and the write controller never access the same logical bank at the same time. This prevents all overflow and underflow conditions.

[0011] In one embodiment, the system includes status flags associated with each data buffer, one flag indicating whether data may be written to the data buffer, and another flag indicating whether data may be read from the
data buffer. The status flags may be stored in the memory device itself or in another memory structure. In an alternative embodiment, a single binary status flag is used to indicate the read/write state of the data buffer.

[0012] In another embodiment, the controller that partitions the memory device into multiple data buffer banks also manages data flowing into and out of the data buffers. The controller may include separate write and read controllers, which may use the status flags discussed above to determine when to write data to and read data from the data buffers.

[0013] Advantages of the invention include one or more of the following:

- Using a single memory device to create multiple data buffers yields an efficient and effective data buffering system that is less expensive than previous systems.
- Data from one or more data transactions may be written to and read from the buffering system in an interleaved, concurrent fashion.
- A single memory device may be used to buffer data between two buses operating at different frequencies without underflowing or overflowing, regardless of a substantial disparity in the buses' operating frequencies.

[0014] The details of one or more embodiments of the invention are set forth in the accompanying drawings and the description below. Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

[0015] FIG. 1 is a block diagram of various components in a computer system.

FIG. 2 is a block diagram of a system for buffering data that flows between data buses operating at different frequencies.

FIG. 3 is a timing diagram for one embodiment of the invention.

Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0016] Referring to FIG. 1, a computer system 10 includes at least two data buses that facilitate data transfers among components of the computer system. One of these data buses, the primary bus 12, is a "host" bus dedicated to carrying out data transactions involving the computer's central processing unit (CPU) 14 and main memory 16. The host bus 12 operates at a first frequency (e.g., 50 MHz). A secondary bus 18 carries out transactions involving other devices 20, 22 in the computer system 10, including transactions between the devices 20, 22 themselves, as well as transactions between one or more of the devices 20, 22 and the CPU 14. The secondary bus, which may be any type of data bus (e.g., ISA, EISA, PCI, SCSI), operates at a second frequency (e.g., 8 MHz). The devices 20, 22 residing on the secondary bus 18 may be any types of devices, including peripheral devices such as a video card or a network interface card (NIC), and input/output (I/O) devices such as a keyboard or a printer.

[0017] The primary bus 12 and the secondary bus 18 are connected by a bridge device 24 which manages data transactions between the two buses. The bridge device 24 may serve both as a translator to interpret commands originating on one bus for delivery to the other bus and as a buffering system for data transfers associated with the commands. The buffering scheme employed by the bridge device 24 is described below. A primary bus controller 26 manages competing requests from the CPU 14 and the bridge device 24 for access to the primary bus 12, in known fashion.

[0018] Likewise, a secondary bus controller 28 manages competing requests from the secondary devices 20, 22 and the bridge device 24 for access to the secondary bus 18.

[0019] Referring to FIG. 2, the bridge device 24 employs a "virtual FIFO" buffering scheme in which a single FIFO memory device 31 is partitioned into N distinct memory regions ("banks") 32-1, 32-2, ..., 32-N by a controller 35. Preferably, each bank comprises a continuous block of memory addresses. The controller 35 treats the banks 32-1, 32-2, ..., 32-N as distinct FIFO buffers, each of which may store data independently of or in conjunction with the other banks. For example, the banks 32-1, 32-2, ..., 32-N may be used individually to buffer data associated with multiple distinct transactions, or they may be used together to buffer data associated with fewer but larger transactions. A bank may range in size from one word (e.g., two bytes) of data to the entire storage capacity of the memory device 31. Therefore, the memory device 31 may contain as few as one bank, or it may contain as many banks as data words that it can store. The trade off is the number of status flags that the read and write controllers need to track, the complexity of the controller, and the efficiency of the FIFO. That is, as N grows bigger, the FIFO efficiency increases, but the number of status flags and the complexity of the controller also increases. In the illustrated embodiment, a memory map 40 created and maintained by the controller 35 identifies the memory addresses that define the boundaries of the banks 32-1, 32-2, ..., 32-N, regardless of how many banks have been created.

[0020] A write controller 42 manages the flow of data into the banks 32-1, 32-2, ..., 32-N, and a read controller 44 manages the flow of data out of the banks. The write and read controllers 42, 44 may be implemented, for example, as state machines. The write and read controllers 42, 44 allow data to be written to one bank and read from the other bank.
another bank in the memory device 31 concurrently. The controllers 42, 44 ensure the integrity of data in the buffers by allowing a device to begin writing data into a bank only when the bank is empty, and to begin reading from a bank only when the bank is full.

[0021] Each bank 32-1, 32-2, ... 32-N has at least two associated status flags: a "done" flag 34-1, 34-2, ... 34-N that indicates to the write controller 42 that the corresponding bank 32-1, 32-2, ... 32-N is empty and can accept data from a writing device; and a "start" flag 36-1, 36-2, ... 36-N that indicates to the read controller 44 that the corresponding bank is full and therefore can provide data to a reading device. The write controller 42 clears the "done" flag for a bank when a writing device begins placing data into the bank. The write controller 42 then sets the bank's "start" flag when the bank is full or, alternatively, when the writing device concludes its data transfer before the bank is full. Likewise, the read controller 44 clears the "start" flag for a bank when a reading device begins taking data from the bank and then sets the "done" flag when the bank is empty. In a preferred embodiment, the "start" and "done" flags are both cleared at the beginning of a data transfer and at "power on". For maximum data integrity, the "start" flag of a bank is also used to clear the "done" flag of that same bank, and the "done" flag is used to clear the "start" flag of the bank. Such an interlock hand-shake mechanism allows the clock rates from both side of the FIFO to be completely independent of each other and there is no limit on the operation range.

[0022] The "start" and "done" flags may be stored in any of several locations, including the following: within the banks 32-1, 32-2, ... 32-N, within the memory device 31 but outside of the banks 32-1, 32-2, ... 32-N, or in another memory structure 38, such as a register (register 38 will not be present if the "start" and "done" flags are stored in the memory device 31). Alternatively, the write and read controllers 42, 44 may keep internal "start" and "done" flags, and each controller 42, 44 may send a message directly to the other controller 44, 42 upon filling or emptying a buffer.

[0023] In an alternative embodiment, a single binary status flag is used to indicate "start" and "done." In this case, the write controller 42 cannot write to a bank until the flag for the bank is set to "done." The write controller 42 then writes to that bank, and, when done writing, resets the corresponding flag to "start." Similarly, the read controller 44 cannot read from a bank until the corresponding flag is reset to "start." After reading all of a bank, the read controller 44 sets the corresponding flag to "done." However, in this embodiment, care should be taken to prevent a "race" condition, where one controller is reading the state of a flag and acting on it while the other controller is changing the flag.

[0024] To ensure that the "virtual FIFO" buffering scheme functions properly regardless of the data transfer rates of the data buses 12, 14, the write controller 42 synchronizes its setting of the "start" flags to the read controller's internal clock (not shown), and the read controller 44 synchronizes its setting of the "done" flags to the write controller's internal clock (not shown). The "start" and "done" flags provide a mechanism that ensures that the read and write controllers 42, 44 never access the same logical bank at the same time. This prevents all overflow and underflow conditions. The "virtual FIFO" buffering scheme provides sufficient buffering despite wide differences in the operating frequencies of the data buses.

[0025] In addition to at least one memory device 31 having multiple data buffer banks, the bridge device 24 may include one or more additional memory devices 46, each of which also may be partitioned as described above. In this situation, memory controller 35 may be used to manage all memory devices in the bridge device 24, or a separate memory controller may be provided for each memory device in the bridge. Each of the memory devices 31, 46 may be embodied in a single random access memory (RAM) integrated circuit. Alternatively, such memory devices 31, 46 may be embodied in a memory module (e.g., a SIMM or DIMM) comprising such integrated circuits and functioning as a single addressable device. In either case, the memory devices 31, 46 may include volatile or non-volatile memory, and may be of any RAM type (e.g., DRAM, SRAM, EDO-RAM, etc.), but is preferably static RAM. The memory devices 31, 46 may be partitioned into banks of equal size, or the banks may be of different sizes, which may be useful, e.g., in applications using asymmetric data transfers. Each memory device may be permanently partitioned once, or the banks may be defined dynamically as needed. The "start" and "done" flags described above typically will be a single bit in length, but flags of other sizes (e.g., nibbles, bytes, words, etc.) also may be used. Alternatively, as noted above, a single bit may be used for each "start" and "done" flag pair (i.e., toggling between high and low logic levels may be used to indicate the "start" and "done" conditions).

[0026] Optionally, each bank may have an associated "last" flag, which indicates that a bank is the last bank for a current data transfer operation. In such a configuration, data is written into the FIFO in the sequential fashion (i.e., bank 32-1 is written until full, then bank 32-2, ..., 32-N, and back to bank 32-1. Data is read out in a similar sequential fashion.

[0027] After a bank is filled, the write controller 42 sets the corresponding "start" flag for the bank. As the last location of a bank is written, the write controller 42 checks the "start" flag of the next bank. If that "start" flag is cleared (the bank is empty), the write controller 42 will continue the write operation. If the "start" flag of the next bank is not cleared (the bank is not empty), the write operation will pause.

[0028] However, the write controller 42 continues to monitor the "start" flag of the next bank. When the "start" flag becomes cleared, the write operation resumes. As the last word of the data transfer is written into a bank, the write controller 42 sets the "start" flag of the bank
(even though the bank may or may not be full); and also sets the "last" flag of the bank.

[0029] The read controller 44 reads from the banks having set "start" flags. After a bank is read out, the read controller 44 sets the "done" flag of that bank, indicating that the bank is available for writing. When the last location of a bank is read, the read controller 44 checks the "start" flag of the next bank. If the "start" flag is set (the bank is ready), the read controller 44 continues the read operation. If the "start" flag is not set (the bank is not ready), the read operation will pause. However, the read controller 44 continues to monitor the "start" flag of the next bank. When the "start" flag becomes set, the read operation resumes. If the current bank has both the "start" and the "last" status flags set, the read controller 44 knows that this is the last bank for the current data transfer operation. Accordingly, the read controller 44 reads data from the bank until the read pointer aligns with the write pointer, and then sets the "done" flag for the bank.

[0030] In an alternative embodiment, the number of bytes or words in each write operation is communicated to the read controller 44 directly. In this configuration, a "last" status flag is not needed.

[0031] FIG. 3 is a timing diagram for one embodiment of the invention. The timing diagram shows a 96-word data transfer using a configuration in which the width of the FIFO memory device 31 is 32 words with a depth of 48 rows, and the device is partitioned into 3 banks. The handshake between the "start" bits and the "done" bits is also shown. In this example, the write clock is twice as fast as the read clock. The following annotations describe the sequence of events:

(W1) After writing to the last location of bank 1, a start 1 flag is set.
(W2) After writing to the last location of bank2, a start2 flag is set.
(W3) After writing to the last location of bank3, a start3 flag is set.
(W4) Setting of start1 causes the read controller to start reading.
(W5) After filling up bank1 for the second time, a done1 flag is reset.
(W6) After filling up bank2 for the second time, a done2 flag is reset.
(W7) After filling up bank3 for the second time, a done3 flag is reset.
(W8) After reading each bank, the read controller checks for the start bit of the next bank. In this case, it is set, so the read continues.
(W9) WREN is disabled (stop writing to FIFO) because after finishing bank3, the write controller notices that start1 is still set (bank1 is not empty). The writing operation continues after start1 is cleared.
(R1) After reading the last location in bank1, done1 is set which resets start1.
(R2) After reading the last location in bank2, done2 is set which resets start2.

(R3) After reading the last location in bank3, done3 is set which resets start3.
(R4) After reading the last location in bank3, the read controller detects that the start bit of the next bank (bank1) is not set, so it stops the read operation.

[0032] A number of embodiments of the present invention have been described. Nevertheless, it will be understood that various modifications may be made without departing from the scope of the invention. For example, while the invention has been described in terms of data transactions involving a host bus, the invention may be used for data transactions between other types of buses, such as PCI-to-PCI bus transactions or PCI-to-I/O bus transactions. Accordingly, other embodiments are within the scope of the following claims.

Claims

1. A first-in-first-out bridge device (24) for managing data transfers between a source bus (12) and a destination bus (18) each operating at a different rate, comprising:

   a memory device (31) dynamically partitionable into a variable number of N
   distinct memory banks (32-1, ..., 32-N), each memory bank storing data to be transferred from
   the source bus (12) to the destination bus (18) during a data transfer operation;
   and

   a controller (35) operable to partition the memory device (31) and including a write controller
   (42) and a read controller (44); said controller (35) characterized in that said write controller (42) and said read controller (44) cooperate to set status flags to allow writing data into a bank only when the bank is empty.

2. The bridge device of claim 1, wherein the bridge device is a single integrated circuit.

3. The bridge device of claim 2, wherein the memory device is dynamically partitioned by the controller to maximize data flow rate from the source bus to the destination bus.

4. The bridge device of claim 3, wherein the controller dynamically partitions the memory device in response to a set of externally input parameters.

5. The bridge device of claim 4, wherein the bridge device is a single integrated circuit and the externally input parameters are programmed inputs to the integrated circuit.

6. The bridge device of claim 5, wherein the externally input parameters includes information about the
7. The bridge device of claim 5, wherein the externally input parameters includes the value of variable number N of the distinct memory banks into which the memory device is to be partitioned.

8. The bridge device of claim 5, wherein the externally input parameters includes information about a desired size of each of the N distinct memory banks.

9. A method of managing data transfers between a source bus (12) and a destination bus (18) each operating at a different rate, in a FIFO bridge device (24) including a controller (35) and a memory device (31), comprising the controller performed steps of:

   dynamically partitioning the memory device (31), in response to a set of externally input parameters, into a variable number of N distinct memory banks (32-1, ..., 32-N), each memory bank for storing data to be transferred during a data transfer operation;

   writing data into a first memory bank of the N distinct memory banks;

   setting a flag when the first memory bank is full;

   writing data into a second memory bank of the N distinct memory banks;

   checking the flag of the first memory bank to determine if it is full;

   the method characterized by initiating a readout of the data stored in the first memory bank only when the flag is set to full.

10. The method of claim 9, wherein the bridge device is a single integrated circuit and the externally input parameters are programmed inputs to the integrated circuit.

11. The method of claim 9, wherein the externally input parameters includes information about the length of each data transfer.

12. The method of claim 9, wherein the externally input parameters includes the value of the variable number N of the distinct memory banks into which the memory device is to be partitioned.

13. The method of claim 9, wherein the externally input parameters includes information about a desired size of each of the N distinct memory banks.

14. The method of claim 9, wherein the memory device is dynamically partitioned to maximize data flow rate from the source bus to the destination bus.

15. The method of claim 9, wherein the step of initiating the readout is performed before the second memory bank becomes full.

16. The method of claim 15, wherein the step of initiating readout is executed simultaneously with the writing of data in the second memory bank.

17. The method of claim 16, further comprising the steps of reading out the data from the first memory bank and setting a done status flag when reading out of data stored therein is complete.

18. The method of claim 16, wherein the controller includes a read controller and a write controller, the method further including the steps of:

   setting, by the write controller, a last status flag after the last word is written into the second memory bank, and

   detecting, by the read controller, the last status flag and reading out the data in the second memory bank after the first memory bank is readout.

**Patentansprüche**

1. Durchlaufspeicher-Brückenelement (24) zur Verwaltung von Datenübertragungen zwischen einem Quellenbus (12) und einem Zielbus (18), die jeweils mit unterschiedlicher Geschwindigkeit arbeiten, wobei der Baustein aufweist:

   einen Speicherbaustein (31), der dynamisch in eine veränderliche Anzahl von N getrennten Speicherbänken (32-1, ..., 32-N) unterteilbar ist, wobei jede Speicherbank Daten speichert, die während eines Datenübertragungsvorganges von dem Quellenbus (12) zum Zielbus (18) zu übertragen sind; und

   eine Steuereinheit (35), die getrieben werden kann, um den Speicherbaustein (31) zu unterteilen, und die eine Schreibsteuereinheit (42) und eine Lesesteuereinheit (44) enthält; wobei die Steuereinheit (35) dadurch gekennzeichnet ist, daß die Schreibsteuereinheit (42) und die Lesesteuereinheit (44) zusammenwirken, um Zustandsflags zu setzen und das Einschreiben von Daten in eine Bank nur dann zulassen, wenn die Bank leer ist.

2. Brückenelement nach Anspruch 1, wobei das Brückenelement ein einziger integrierter Schaltkreis ist.

3. Brückenelement nach Anspruch 2, wobei der Speicherbaustein durch die Steuereinheit dynamisch unterteilt wird, um die Datenflußgeschwindigkeit vom Quellenbus zum Zielbus zu maximieren.

4. Brückenelement nach Anspruch 3, wobei die Steu-
ereinheit als Reaktion auf einen Satz von extern eingegebenen Parametern den Speicherbaustein dynamisch unterteilt.

5. Brückenelement nach Anspruch 4, wobei das Brückenelement ein einziger integrierter Schaltkreis ist und die extern eingegebenen Parameter programmierte Eingaben in den integrierten Schaltkreis sind.

6. Brückenelement nach Anspruch 5, wobei die extern eingegebenen Parameter Informationen über die Länge jeder Datenübertragung enthalten.

7. Brückenelement nach Anspruch 5, wobei die extern eingegebenen Parameter den Wert der variablen Anzahl N der getrennten Speicherbänke enthalten, in die der Speicherbaustein zu unterteilen ist.

8. Brückenelement nach Anspruch 5, wobei die extern eingegebenen Parameter Informationen über eine gewünschte Größe jeder von den N getrennten Speicherbänke enthalten.

9. Verfahren zum Verwalten von Datenübertragungen zwischen einem Quellenbus (12) und einem Zielbus (18), die jeweils mit unterschiedlicher Geschwindigkeit arbeiten, in einem FIFO-Brückenelement (24), das eine Steuereinheit (35) und einen Speicherbaustein (31) enthält, wobei das Verfahren die folgenden, durch die Steuereinheit ausgeführten Schritte aufweist:

dynamische Unterteilung des Speicherbausteins (31) als Reaktion auf einen Satz von extern eingegebenen Parametern in eine variable Anzahl von N getrennten Speicherbänken (32-1, ..., 32-N), wobei jede Speicherbank dazu dient, während einer Datenübertragungsoperation zu übertragende Daten zu speichern; Schreiben von Daten in eine erste Speicherbank von den N getrennten Speicherbänken; Setzen eines Flags, wenn die erste Speicherbank voll ist; Schreiben von Daten in eine zweite Speicherbank von den N getrennten Speicherbänken; Prüfen des Flags der ersten Speicherbank, um festzustellen, ob sie voll ist; wobei das Verfahren dadurch gekennzeichnet ist, daß ein Auslesen der in der ersten Speicherbank gespeicherten Daten erst dann eingeleitet wird, wenn das Flag auf "voll" gesetzt ist.

10. Verfahren nach Anspruch 9, wobei das Brückenelement ein einziger integrierter Schaltkreis ist und die extern eingegebenen Parameter programmierte Eingaben in den integrierten Schaltkreis sind.

11. Verfahren nach Anspruch 9, wobei die extern eingegeben Parameter Informationen über die Länge jeder Datenübertragung enthalten.

12. Verfahren nach Anspruch 9, wobei die extern eingegebenen Parameter den Wert der variablen Anzahl N der getrennten Speicherbänke enthalten, in die der Speicherbaustein zu unterteilen ist.

13. Verfahren nach Anspruch 9, wobei die extern eingegebenen Parameter Informationen über eine gewünschte Größe jeder von den N getrennten Speicherbänke enthalten.

14. Verfahren nach Anspruch 9, wobei der Speicherbaustein dynamisch unterteilt wird, um die Datenflußgeschwindigkeit vom Quellenbus zum Zielbus zu maximieren.

15. Verfahren nach Anspruch 9, wobei der Schritt zur Einleitung des Auslesevorgangs durchgeführt wird, bevor die zweite Speicherbank voll wird.


17. Verfahren nach Anspruch 16, das ferner die Schritte zum Auslesen der Daten aus der ersten Speicherbank und zum Setzen eines "Fertig"-Zustandsflags aufweist, wenn das Auslesen von darin gespeicherten Daten beendet ist.

18. Verfahren nach Anspruch 16, wobei die Steuereinheit eine Lesesteuereinheit und eine Schreibsteuereinheit enthält, wobei das Verfahren ferner die folgenden Schritte aufweist:

Setzen eines "Letzte"-Zustandsflags durch die Schreibsteuereinheit, nachdem das letzte Wort in die zweite Speicherbank eingeschrieben ist, und Erfassen des "Letzte"-Zustandsflags durch die Lesesteuereinheit und Auslesen der Daten in der zweiten Speicherbank, nachdem die erste Speicherbank ausgelesen ist.

Revendications

1. Dispositif de pont premier entré, premier sorti (24) pour gérer des transferts de données entre un bus source (12) et un bus destination (18), chacun fonctionnant à une vitesse différente, comprenant :

un dispositif de mémoire (31) partitionnable de manière dynamique en un nombre variable de N blocs de mémoire distincts (32-1, ..., 32-N),
chaque bloc de mémoire stockant des données à transférer du bus source (12) au bus destination (18) pendant une opération de transfert de données, et un contrôleur (35) qui peut être mis en service pour partitionner le dispositif de mémoire (31) et comprend un contrôleur d'écriture (42) et un contrôleur de lecture (44), ledit contrôleur (35) étant caractérisé en ce que ledit contrôleur d'écriture (42) et ledit contrôleur de lecture (44) coopèrent pour placer des drapeaux d'état en vue de permettre une écriture de données dans un bloc uniquement lorsque le bloc est vide.

2. Dispositif de pont suivant la revendication 1, dans lequel le dispositif de pont est un circuit intégré unique.

3. Dispositif de pont suivant la revendication 2, dans lequel le dispositif de mémoire est partitionné de manière dynamique par le contrôleur pour maximiser le débit de données depuis le bus source vers le bus destination.

4. Dispositif de pont suivant la revendication 3, dans lequel le contrôleur partitionne de manière dynamique le dispositif de mémoire en réponse à un ensemble de paramètres d'entrée fournis de l'extérieur.

5. Dispositif de pont suivant la revendication 4, dans lequel les paramètres d'entrée fournis de l'extérieur sont des entrées programmées vers le circuit intégré.

6. Dispositif de pont suivant la revendication 5, dans lequel les paramètres d'entrée fournis de l'extérieur comprennent une information au sujet de la longueur de chaque transfert de données.

7. Dispositif de pont suivant la revendication 6, dans lequel les paramètres d'entrée fournis de l'extérieur comprennent la valeur du nombre variable N des blocs de mémoire distincts dans lesquels le dispositif de mémoire est à partitionner.

8. Dispositif de pont suivant la revendication 5, dans lequel les paramètres fournis de l'extérieur comprennent une information au sujet d'une taille souhaitée de chacun des N blocs de mémoire distinctifs.

9. Procédé de gestion de transferts de données entre un bus source (12) et un bus destination (18) qui fonctionnent chacun à une vitesse différente, dans un dispositif de pont premier entré, premier sorti (24) comportant un contrôleur (35) et un dispositif de mémoire (31), comprenant les étapes effectuées par le contrôleur :

de partition dynamique du dispositif de mémoire (31) en réponse à un ensemble de paramètres d'entrée fournis de l'extérieur, en un nombre variable de N blocs de mémoire distincts (32-1, ..., 32-N), chaque bloc de mémoire étant prévu pour stocker des données à transférer pendant une opération de transfert de données, d'écriture de données dans un premier bloc de mémoire des N blocs de mémoire distincts, de placement d'un drapeau lorsque le premier bloc de mémoire est plein, d'écriture de données dans un second bloc de mémoire des N blocs de mémoire distincts, de vérification du drapeau du premier bloc de mémoire pour déterminer s'il est plein, le procédé étant caractérisé par un démarrage d'une lecture des données stockées dans le premier bloc de mémoire uniquement lorsque le drapeau est placé sur plein.

10. Procédé suivant la revendication 9, dans lequel le dispositif de pont est un circuit intégré unique et les paramètres d'entrée fournis de l'extérieur sont des entrées programmées vers le circuit intégré.

11. Procédé suivant la revendication 9, dans lequel les paramètres d'entrée fournis de l'extérieur comprennent une information au sujet de la longueur de chaque transfert de données.

12. Procédé suivant la revendication 9, dans lequel les paramètres d'entrée fournis de l'extérieur comprennent la valeur du nombre variable N des blocs de mémoire distincts dans lesquels le dispositif de mémoire est à partitionner.

13. Procédé suivant la revendication 9, dans lequel les paramètres d'entrée fournis de l'extérieur comprennent une information au sujet d'une taille souhaitée de chacun des N blocs de mémoire distincts.

14. Procédé suivant la revendication 9, dans lequel le dispositif de mémoire est partitionné de manière dynamique pour maximiser le débit des données depuis le bus source vers le bus destination.

15. Procédé suivant la revendication 9, dans lequel l'étape de démarrage de la lecture est effectuée avant que le second bloc de mémoire ne devienne plein.

16. Procédé suivant la revendication 15, dans lequel l'étape de démarrage de la lecture est exécutée simultanément à l'écriture de données dans le second bloc de mémoire.
17. Procédé suivant la revendication 16, comprenant en outre les étapes de lecture des données du premier bloc de mémoire et de placement d’un drapeau d’état fait lorsque la lecture des données stockées dedans est achevée.

18. Procédé suivant la revendication 16, dans lequel le contrôleur comprend un contrôleur de lecture et un contrôleur d’écriture, le procédé comprenant en outre les étapes :

de placement par le contrôleur d’écriture d’un drapeau d’état dernier après que le dernier mot est écrit dans le second bloc de mémoire, et de détection, par le contrôleur de lecture, du drapeau d’état dernier et de lecture des données dans le second bloc de mémoire après que le premier bloc de mémoire est lu.