US 20080163035A1

(19) United States
(12) Patent Application Publication
Kottke
(10) Pub. No.: US 2008/0163035 A1
(43) Pub. Date: Jul. 3, 2008

(54) METHOD FOR DATA DISTRIBUTION AND DATA DISTRIBUTION UNIT IN A MULTIPROCESSOR SYSTEM

(75) Inventor: **Thomas Kottke**, Ehningen (DE)

Correspondence Address:
KENYON & KENYON LLP
ONE BROADWAY
NEW YORK, NY 10004

(73) Assignee: **ROBERT BOSCH GMBH**, Stuttgart (DE)

(21) Appl. No.: **11/666,406**

(22) PCT Filed: **Oct. 25, 2005**

(86) PCT No.: **PCT/EP05/55532**

§ 371 (c)(1), (2), (4) Date: **Mar. 10, 2008**

(30) **Foreign Application Priority Data**

Oct. 25, 2004	(DE)	10 2004 051 937.4
Oct. 25, 2004	(DE)	10 2004 051 950.1
Oct. 25, 2004	(DE)	10 2004 051 952.8
Oct. 25, 2004	(DE)	10 2004 051 964.1
Oct. 25, 2004	(DE)	10 2004 051 992.7

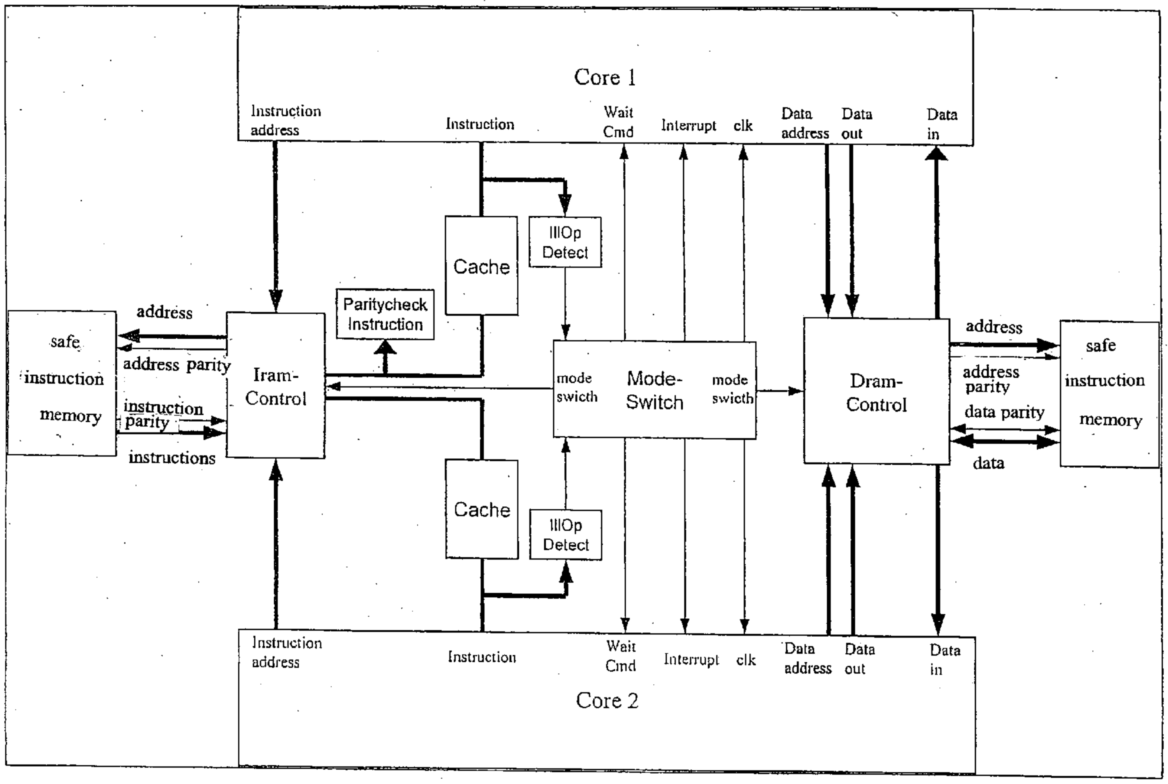
Publication Classification

(51) **Int. Cl.**
G06F 9/318 (2006.01)
G06F 11/08 (2006.01)

(52) **U.S. Cl.** **714/799**; 712/30; 714/E11.03; 712/E09.035

(57) **ABSTRACT**

A unit and method for distributing data from at least one data source in a system provided with at least two computer units, containing switching means which are used to switch between at least two operating modes of the system, wherein data distribution and/or selection of a data source is dependent upon the operating mode.



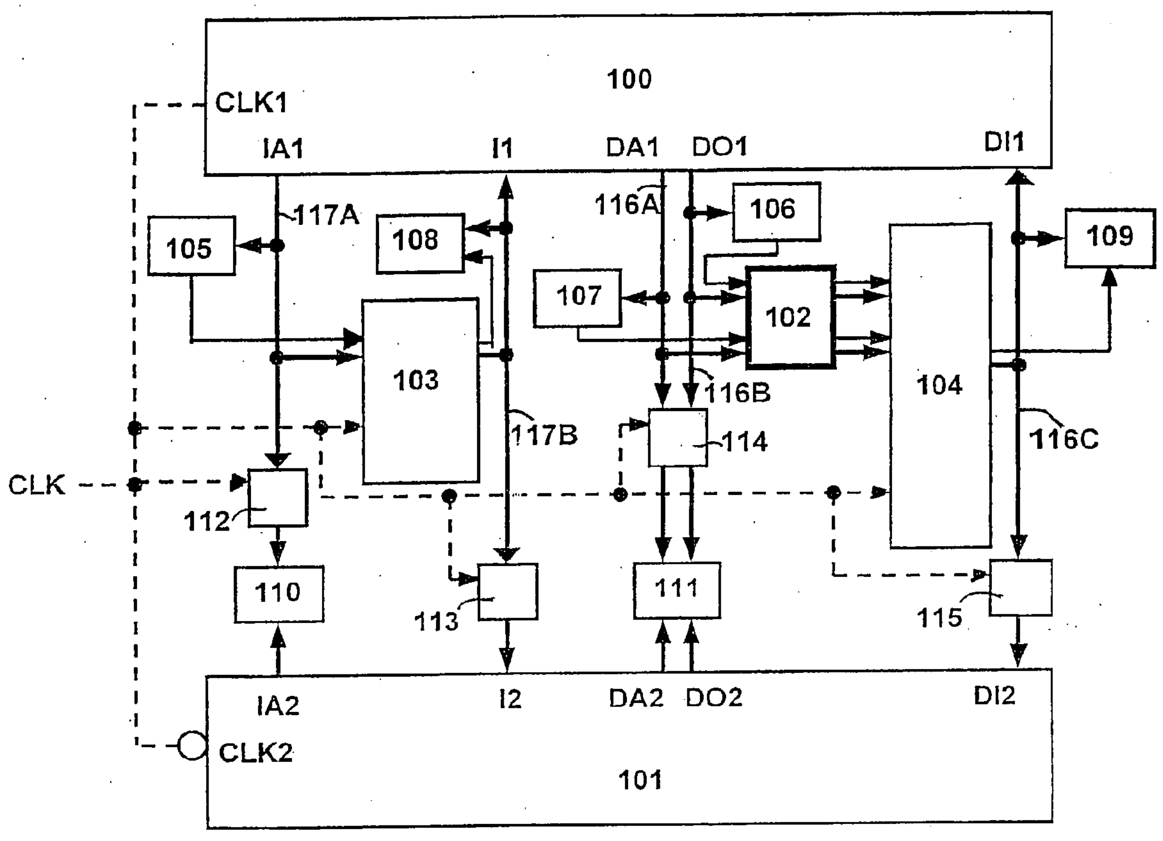

100
CLK1
IA1
I1
DA1
DO1
DI1
117A
105
108
116A
106
109
107
102
103
104
116B
117B
114
116C
CLK
112
110
113
111
115
IA2
I2
DA2
DO2
DI2
CLK2
101

Fig. 1

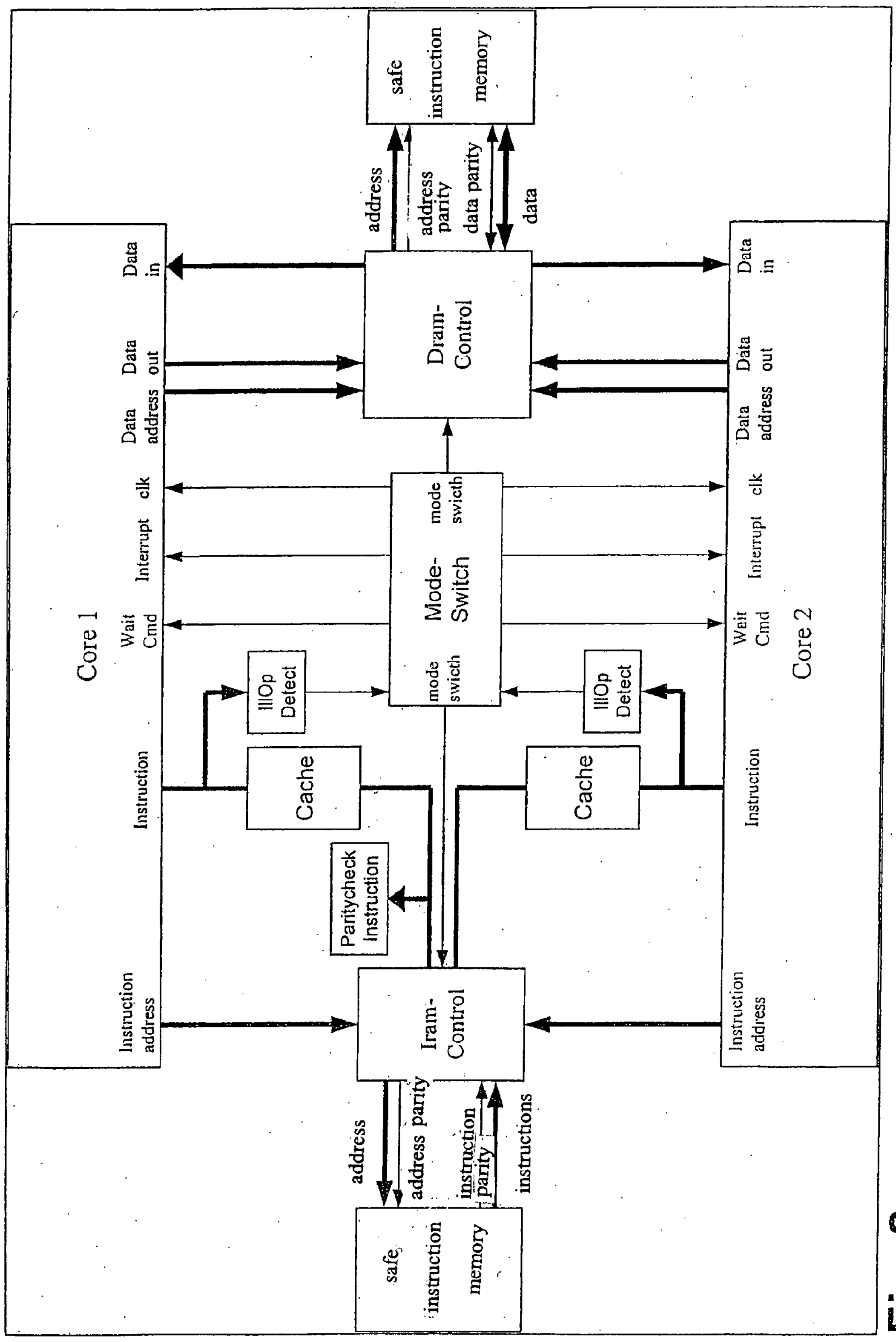

safe instruction memory
address
address parity
data parity
data
Dram-Control
Data in
Data out
Data address
clk
Interrupt
Wait Cmd
Core 1
Core 2
mode switch
Mode-Switch
IllOp Detect
Cache
Instruction
Paritycheck Instruction
Instruction address
Iram-Control
address
address parity
instruction parity
instructions
safe instruction memory

Fig. 2

METHOD FOR DATA DISTRIBUTION AND DATA DISTRIBUTION UNIT IN A MULTIPROCESSOR SYSTEM

BACKGROUND OF THE INVENTION

[0001] 1. Field of the Invention

[0002] The present invention relates to a device and a method for data distribution from at least one data source in a multiprocessor system.

[0003] 2. Description of Related Art

[0004] In technical applications such as in the automobile industry or in the industrial goods industry in particular, i.e., in mechanical engineering and automation, more and more microprocessor-based or computer-based control and regulating systems are being used for applications critical with regard to safety. Dual computer systems or dual processor systems (dual cores) are nowadays widely used computer systems for applications critical with regard to safety, in particular in vehicles, such as antilock systems, electronic stability programs (ESP), X-by-wire systems such as drive-by-wire or steer-by-wire or brake-by-wire, etc. or also in other networked systems. To satisfy these high safety requirements in future applications, powerful error detection mechanisms and error handling mechanisms are needed, in particular to counteract transient errors arising, for example, when the size of semiconductor structures of computer systems is reduced. It is relatively difficult to protect the core itself, i.e., the processor. One approach, as mentioned above, is the use of a dual-core system for error detection.

[0005] Such processor units having at least two integrated execution units are known as dual core or multicore architectures. Such dual core or multicore architectures are currently proposed mainly for two reasons:

[0006] First, they may contribute to an enhanced performance in that the two execution units or cores are considered and treated as two processing units on a single semiconductor module. In this configuration, the two execution units or cores process different programs or tasks. This allows enhanced performance; for this reason, this configuration is referred to as performance mode.

[0007] The second reason for using a dual-core or multicore architecture is enhanced reliability in that the two execution units redundantly process the same program. The results of the two execution units or CPUs, i.e., cores, are compared, and an error may be detected from the comparison for agreement. In the following, this configuration is referred to as safety mode or error detection mode.

[0008] Thus, currently there are both dual processor and multiprocessor systems that work redundantly to recognize hardware errors (see dual core or master checker systems), and dual processor and multiprocessor systems that process different data on their processors. If these two operating modes are combined according to an embodiment of the present invention in a dual processor or multiprocessor system (for the sake of simplicity we shall only refer to dual processor systems; however, the present invention is also applicable to multiprocessor systems), both processors must contain different data in performance mode and the same data in error detection mode.

[0009] An object of the present invention is to provide a unit and a method which delivers the instructions/data to the at least two processors redundantly or differently, depending on the mode, and divides up the memory access rights, in particular in the performance mode.

BRIEF SUMMARY OF THE INVENTION

[0010] Such a unit makes it possible to operate a dual processor system effectively in such a way that switchover during operation is possible in both safety and performance modes. We shall therefore refer to processors, which, however, also includes the concept of cores or execution units.

[0011] The present invention provides a unit for data distribution from at least one data source in a system having at least two execution units and contains switchover means (ModeSwitch) which make switchover between at least two operating modes of the system possible, the unit being designed in such a way that the data distribution and/or the data source depends on the operating mode. A system having such a unit is also presented.

[0012] The present invention also presents a corresponding data distribution method from at least one data source in a system having at least two execution units, which has switchover means which make switchover between at least two operating modes of the system possible, the data distribution and/or a selection of a data source (instruction memory, data memory, cache in particular) depending on the operating mode.

[0013] The first operating mode corresponds to a safety mode in which the two processing units process the same programs and/or data, and comparison means are provided, which compare the states resulting from the processing of the same programs for agreement.

[0014] The unit according to the present invention and the method according to the present invention make optimized implementation of both modes possible in a dual-processor system.

[0015] If the two processors operate in error detection mode (F mode), the two processors receive the same data/instructions; if they operate in performance mode (P mode), each processor may access the memory. In that case, this unit manages the accesses to the single memory or peripheral present.

[0016] In the F mode, the unit receives the data/addresses of a processor (here referred to as “master”) and relays them to the components such as memories, bus, etc. The second processor (here “slave”) intends to access the same device. The data distribution unit receives this request at a second port, but does not relay it to the other components. The data distribution unit transmits the same data to both slave and master and compares the data of the two processors. If they are different, the data distribution unit (here DDU) indicates this via an error signal. Therefore, only the master operates the bus/memory and the slave receives the same data (operating mode as in the case of a dual-core system).

[0017] In the P mode both processors process different program portions. The memory accesses are therefore also different. The DDU therefore receives the request of the processors and returns the results/requested data to the processor that requested them. If both processors intend to access the same component at the same time, one processor is set to a wait state until the other one has been served.

[0018] Switchover between the two modes and thus between the different types of operation of the data distribution unit takes place via a control signal, which may be generated by one of the two processors or externally.

[0019] Switchover is advantageously triggered and/or indicated by a control signal, a mode signal in particular, which refers to the operating mode of at least one processing unit, the control signal being generated externally in particular with reference to the processing units.

[0020] It is furthermore advantageous if switchover is triggered and/or indicated by an instruction, e.g., an instruction which describes an illegal operation (illOp), the instruction being generated by the switchover means, the mode switch unit in particular.

[0021] Input data of both processing units are advantageously compared for agreement in an operating mode which corresponds to a safety mode (F mode) and/or also output data of both processing units are compared for agreement in an operating mode which corresponds to a safety mode (F mode).

[0022] The data to be distributed are advantageously relayed to at least one additional component, a processing unit for example, the data to be distributed being extended by an error detection code prior to relaying. The input data may also be relayed to at least one additional component, a processing unit in particular, the input data being extended by an error detection code prior to relaying. The output data may also be relayed to at least one additional component, the output data being extended by an error detection code prior to relaying. For all these cases, an error signal is advantageously output if an error is detected thanks to the error detection code. In one embodiment, an error signal is output only in the safety mode (F mode).

[0023] Basically, a distinction may be made between a performance mode and a safety mode, and in the performance mode the data of both processing units are prioritized, and this data may be received and/or relayed sequentially as a function of the prioritization.

[0024] A delay component, which delays the preceding data by a clock pulse offset as a function of the clock pulse offset between the two processing units in the particular operating mode, may be included according to the present invention.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

[0025] FIG. **1** shows a schematic illustration of a dual-core computer system.

[0026] FIG. **2** shows an example embodiment of the data distribution unit according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0027] The data to be distributed are advantageously read from a memory and then distributed to the processing units.

[0028] Data distribution is advantageously controlled by state machines, two state machines being provided for each processing unit. They are advantageously configured as one synchronous state machine and one asynchronous state machine.

[0029] A system is provided with such a unit according to the present invention, a monitoring circuit external to the unit being also provided, which detects an error if an intended switchover between the operating modes does not take place.

[0030] If the dual-processor system is operated with a clock pulse offset in the F mode, but not in the P mode, the DDU unit delays the data for the slave as needed, i.e., it stores the master's output data until they may be compared to the slave's output data for error detection.

[0031] The clock pulse offset is elucidated in more detail with reference to FIG. **1**.

[0032] FIG. **1** shows a dual-core system having a first computer **100**, in particular a master computer and a second computer **101**, in particular a slave computer. The entire system is operated at a predefinable clock pulse, i.e., in predefinable clock cycles CLK. The clock pulse is supplied to the computers via clock input CLK**1** of computer **100** and clock input CLK**2** of computer **101**. In this dual-core system, there is also a special feature for error detection in that first computer **100** and second computer **101** operate at a predefinable time offset or a predefinable clock pulse offset. Any desired time period may be defined for a time offset, and also any desired clock pulse regarding an offset of the clock pulses. This may be an offset by an integral number of clock pulses, but also, as shown in this example, an offset by 1.5 clock pulses, first computer **100** working, i.e., being operated here 1.5 clock pulses ahead of second computer **101**. This offset may prevent common mode failures from interfering with the computers or processors, i.e., the cores of the dual-core system, in the same way and thus from remaining undetected. In other words, due to the offset, such common mode failures affect the computers at different points in time during the program run and thus have different effects for the two computers, which makes errors detectable. Under certain circumstances, effects of errors of the same type would not be detectable in a comparison without a clock pulse offset; this is avoided by the method according to the present invention. To implement this time or clock pulse offset, 1.5 clock pulses in this particular case of a dual-core system, offset modules **112** through **115** are provided.

[0033] To detect the above-mentioned common mode errors, this system is designed to operate at a predefined time offset or clock pulse offset, here of 1.5 clock pulses, i.e., while one of the computers, e.g., computer **100**, is directly addressing external components **103** and **104** in particular, second computer **101** is running with a delay of exactly 1.5 clock pulses. To generate the desired 1.5-pulse delay in this case, computer **101** is supplied with the inverted clock signal at clock input CLK**2**. However, the above-mentioned terminals of the computer, i.e., its data and/or instructions, must therefore also be delayed by the above-mentioned clock pulses, here 1.5 clock pulses in particular; as mentioned previously offset or delay modules **112** through **115** are provided for this purpose. In addition to the two computers or processors **100** and **101**, components **103** and **104** are provided, which are connected to the two computers **100** and **101** via bus **116**, having bus lines **116**A, **116**B, and **116**C, and bus **117**, having bus lines **117**A and **117**B. Bus **117** is an instruction bus, **117**A being an instruction address bus and **117**B being the partial instruction (data) bus. Address bus **117**A is connected to computer **100** via an instruction address terminal IA**1** (instruction address **1**) and to computer **101** via an instruction address terminal IA**2** (instruction address **2**). The instructions proper are transmitted via partial instruction bus **117**B, which is connected to computer **100** via an instruction terminal I**1** (instruction **1**) and to computer **101** via an instruction terminal I**2** (instruction **2**). In this instruction bus **117** having **117**A and **117**B, one component **103**, an instruction memory, for example, a safe instruction memory in particular or the like, is connected in between. This component, in particular as an instruction memory, is also operated at clock rate CLK in this example. In addition, a data bus **116** has a data address bus or data address line **116**A and a data bus or data line **116**B. Data address bus or data address line **116**A is connected to computer **100** via a data address terminal DA**1** (data address **1**) and to computer **101** via a data address terminal DA**2** (data address **2**). Also data bus or data line **116**B is connected to computer **100** via a data terminal DO**1** (data out **1**) and to computer **101** via a data terminal DO**2** (data out **2**). Furthermore, data bus **116** has data bus line **116**C, which is connected to computer **100** via a data terminal DI**1** (data in **1**) and to computer **101** via a data terminal DI**2** (data in **2**). In this data bus **116** having lines **116**A, **116**B, and **116**C, a component **104**, a data memory for example, a safe data memory in particular or the like, is connected in between. This component **104** is also supplied with clock cycle CLK.

[0034] Components **103** and **104** represent any components that are connected to the computers of the dual-core system via a data bus and/or instruction bus and are able to receive or output erroneous data and/or instructions corresponding to accesses via data and/or instructions of the dual-core system for read and/or write operations. Error identifier generators **105**, **106**, and **107**, which generate an error identifier such as a parity bit, or another error code such as an error correction code (ECC), or the like, are provided for error prevention. For this purpose, appropriate error identifier checking devices **108** and **109** are also provided for checking the particular error identifier, i.e., the parity bit or another error code such as ECC, for example.

[0035] In the redundant design in the dual-core system, the data and/or instructions are compared in comparators **110** and **111** as depicted in FIG. **1**. However, if there is a time offset, a clock pulse offset for example, between computers **100** and **101**, caused either by a non-synchronous dual-core system or, in the case of a synchronous dual-core system by synchronization errors, or as in this special example, by a time or clock pulse offset, here of 1.5 clock pulses in particular, provided for error detection, a computer, computer **100** in particular in this case, may write or read erroneous data and/or instructions into components, external components in particular such as memory **103** or **104** in particular in this case, but also with regard to other users or actuators or sensors during this time or clock pulse offset. It may thus erroneously perform a write access instead of an intended read access due to this clock pulse offset. These scenarios result, of course, in errors in the entire system, in particular without a clear possibility to display which data and/or instructions exactly have been erroneously changed, which also causes recovery problems.

[0036] In order to eliminate this problem, a delay unit **102**, as shown, is connected into the lines of the data bus and/or into the instruction bus. For the sake of clarity, only connection into the data bus is depicted. Of course, connection into the instruction bus is also possible and conceivable. This delay unit **102** delays the accesses, the memory accesses in particular in this case, so that a possible time offset or clock pulse offset is compensated, in particular in the case of an error detection, for example, via comparators **110** and **111**, at least until the error signal is generated in the dual-core system, i.e., the error is detected in the dual-core system. Different variants may be implemented:

[0037] Delay of the write and read operations, delay of the write operations only, or, although not preferably, delay of the read operations. A delayed write operation may then be converted into a read operation via a change signal, e.g., the error signal, in order to avoid erroneous writing.

[0038] An exemplary implementation of the data distribution unit (DDU) which has a device for detecting the switchover intent (via IllOPDetect), since the IllOP instruction (IllOP=illegal operation) is used for switchover in this example, the mode switch unit, and the iram and dram control module is explained with reference to FIG. **2**.

[0039] IllOpDetect: Switchover between the two modes is detected by the “switch detect” units. The unit is situated between the cache and the processor on the instruction bus and shows whether the IllOp instruction is loaded into the processor. If the instruction is detected, this event is communicated to the mode switch unit. The switch detect unit is provided separately for each processor. The switch detect unit does not have to have an error-tolerant design, since it is present in duplicate, i.e., redundantly. It is also conceivable to design this unit to be error-tolerant and thus without redundancy.

[0040] ModeSwitch: Switchover between the two modes is triggered by the “switch detect” unit. If a switchover is to be performed from lock mode to split mode, both switch detect units detect the switchover, since both processors are processing the same program code in the lock mode. The switch detect unit of processor **1** detects these 1.5 clock pulses before the switch detect unit of processor **2**. The mode switch unit stops processor **1** for two pulses with the aid of the wait signal. Processor **2** is also stopped 1.5 clock pulses later, but only for one-half of a clock pulse, thus being synchronized to the system clock. The status signal is subsequently switched to split for the other components, and the two processors continue to operate. For the two processors to execute different tasks, they must diverge in the program code. This takes place via a read access to the processor ID directly after switching over into the split mode. The processor ID read is different for each of the two processors. If a comparison is to be made with a reference processor ID, the corresponding processor may be brought to another program point using a conditional jump instruction. When switching over from split mode to lock mode, this is noticed by a processor, i.e., by one before the other. This processor will execute program code containing the switchover instruction. This is now registered by the switch detect unit, which informs the mode switch unit accordingly. The mode switch unit stops the corresponding processor and informs the second one of the synchronization intent via an interrupt. The second processor receives an interrupt and may now execute a software routine to terminate its task. It then jumps to the program point where the switchover instruction is located. Its switch detect unit now also signals the intent to change modes to the mode switch unit. At the next rising system clock edge, the wait signal is deactivated for processor **1** and, 1.5 clock pulses later, for processor **2**. Now both processors work synchronously with a clock pulse offset of 1.5 clock pulses.

[0041] If the system is in lock mode, both switch detect units must inform the mode switch unit that they intend to switch to the split mode. If the switchover intent is only communicated by one unit, the error is detected by the comparator units, since these continue to receive data from one of the two processors, and these data are different from that of the stopped processor.

[0042] If both processors are in the split mode and one does not switch back to the lock mode, this may be detected by an external watchdog. In the event of a trigger signal for each processor, the watchdog notices that the waiting processor is no longer sending messages. If there is only one watchdog signal for the processor system, the watchdog may only be triggered in the lock mode. The watchdog would thus detect that no mode switchover has taken place. The mode signal is in the form of a dual-rail signal, where 10 stands for the lock mode and 01 for the split mode. 00 and 11 indicate errors.

[0043] IramControl: Access to the instruction memory of both processors is controlled via the IRAM control, which must have a reliable design, since it is a single point of failure. It has two state machines for each processor: a synchronous state machine iram**1**clkreset and an asynchronous state machine readiram**1**. In the safety-critical mode, the state machines of the two processors monitor one another, and in the performance mode they operate separately.

[0044] Reloading of the two caches of the processors is controlled by two state machines, one synchronous state machine iramclkreset and an asynchronous state machine readiram. These two state machines divide the memory accesses in the split mode. Processor **1** has the higher priority. After an access to the main memory by processor **1**, if both processors now intend to access the main memory, processor **2** receives the memory access permission. These two state machines are implemented for each processor. In the lock mode, the output signals of the state machines are compared in order to detect the occurrence of any error.

[0045] The data for updating cache **2** in the lock mode are delayed by 1.5 clock pulses in the IRAM control unit.

[0046] In bit **5** in register **0** of SysControl, the identity of the core is encoded. In the case of core **1** the bit is 0 and in the case of core **2** it is high. This register is mirrored in the memory area having the address 65528.

[0047] In the event of a memory access by core **2**, a check is first made to determine in what mode the core is operating. If it is in the lock mode, its memory access is suppressed. This signal is in the form of a common rail signal, since it is critical with regard to safety.

[0048] The program counter of processor **1** is delayed by 3.5 clock pulses to enable a comparison with the program counter of processor **2** in the lock mode.

[0049] In the split mode, the caches of both processors may be reloaded separately. If a switchover into the lock mode is performed, the two caches are not coherent with respect to one another. This may cause the two processors to diverge and the comparators to thus signal an error. To avoid this, a flag table is constructed in the IRAM control, where it is noted whether a cache line has been written in the lock mode or in the split mode. When the cache is reloaded in the lock mode, the entry corresponding to the cache line is set at 0, and when it is reloaded in the split mode or when the cache line of a single cache is updated, it is set at 1. If the processor now accesses the memory in the lock mode, a check is performed of whether this cache line has been updated in the lock mode, i.e., whether it is identical in the two caches. In the split mode, the processor may always access the cache line, regardless of the status of the Flag Vector. This table must be present only once, since in the event of an error, the two processors diverge and thus this error is reliably detected by the comparators. Since the access times to the central table are relatively long, this table may also be copied to each cache.

[0050] DramControl: The parity is formed in this component for the address, data, and memory control signals of each processor.

[0051] There is a process for both processors for locking the memory. This process does not have to have a fail-safe design, since in the lock mode erroneous memory accesses are detected by the comparators and in the split mode no safety-relevant applications are executed. A check is performed here of whether the processor intends to lock the memory for the other processor. The data memory is locked via an access to the memory address $FBFF$=64511. This signal must be applied for one cycle even if a wait instruction is being applied to the processor at the time of the call. The state machine for managing the data memory access has two main states:

[0052] processor status lock: Both processors operate in the lock mode. This means that the data memory locking function is not needed. Processor **1** coordinates the memory accesses.

[0053] processor status split: A data memory access conflict resolution is now necessary, and memory lock must be able to occur.

[0054] The split mode state is in turn subdivided into seven states which resolve the access conflicts and are able to lock the data memory for the other processor. When both processors intend to access the memory at the same time, the order of execution represents the priorities at the same time.

[0055] Core**1**_Lock: Processor **1** has locked the data memory. If processor **2** intends to access the memory in this state, it is stopped by a wait signal until processor **1** releases the data memory again.

[0056] Core**2**_Lock: This is the same state as the previous one, except that now processor **2** has locked the data memory and processor **1** is stopped for data memory operations.

[0057] lock**1**_wait: The data memory was locked by processor **2** as processor **1** also intended to reserve it for itself. Processor **1** is thus pre-marked for the next memory lock.

[0058] nex: The same for processor **2**. The data memory was locked by processor **1** during the locking attempt. The memory is pre-reserved for processor **2**. In the event of normal memory access without locking, processor **2** may have access before processor **1** if processor **1** was up previously.

[0059] Memory access by processor **1**: The memory is not locked in this case. Processor **1** is allowed to access the data memory. If it intends to lock it, it may do so in this state.

[0060] Memory access by processor **2**. Processor **1** did not intend to access the memory in the same clock pulse; therefore, the memory is free for processor **2**.

[0061] No processor intends to access the data memory.

[0062] As mentioned previously, the DDU has the switchover intent detector (IllOPDetect), the mode switch unit, and the Iram and Dram control.

[0063] As explained previously, the core of the invention is the general mode of operation of the data distribution unit DDU (different data assignment and thus also operating mode selection, depending on the mode).

1-27. (canceled)

28. A method for data distribution from at least one data source in a system having at least two processing units, the method comprising:

providing a switchover unit which implements a switchover between at least two operating modes of the system; and

performing the data distribution, wherein at least one of the data distribution and a selection of a data source depends on an operating mode of the system.

29. The method as recited in claim **28**, wherein the switchover is at least one of triggered and indicated by a control signal that refers to the operating mode of at least one processing unit.

30. The method as recited in claim **29**, wherein the control signal is generated externally with respect to the processing units.

31. The method as recited in claim **28**, wherein the switchover is at least one of triggered and indicated by an instruction describing an illegal action.

32. The method as recited in claim **31**, wherein the instruction is generated by the switchover unit.

33. The method as recited in claim **28**, further comprising: comparing, in an operating mode corresponding to a safety mode, input data of the at least two processing units with respect to one another for agreement.

34. The method as recited in claim **28**, further comprising: comparing, in an operating mode corresponding to a safety mode, output data of the at least two processing units with respect to one another for agreement.

35. The method as recited in claim **28**, further comprising: relaying the data to be distributed, wherein the data to be distributed are relayed to a processing unit, and wherein the data to be distributed are extended by an error detection code prior to the relaying.

36. The method as recited in claim **33**, further comprising: relaying the input data to a processing unit, wherein the input data are extended by an error detection code prior to the relaying.

37. The method as recited in claim **34**, further comprising: relaying the output data to at least one additional component, wherein the output data are extended by an error detection code prior to the relaying.

38. The method as recited in claim **33**, further comprising: outputting an error signal in the event of non-agreement.

39. The method as recited in claims **35**, further comprising: outputting an error signal if an error is detected on the basis of the error detection code.

40. The method as recited in claim **38**, wherein the error signal is output only in the safety mode.

41. The method as recited in claim **28**, wherein the at least two operating modes of the system include a performance mode and a safety mode, and wherein in the performance mode the data of both processing units are prioritized and at least one of sequentially received and relayed as a function of assigned priorities.

42. A device for performing data distribution from at least one data source in a system having at least two processing units, comprising:

a switchover unit which implements a switchover between at least two operating modes of the system;

wherein at least one of the data distribution and a selection of a data source depends on an operating mode of the system.

43. The device as recited in claim **42**, further comprising: a comparison unit;

wherein, in a first operating mode corresponding to a safety mode in which the at least two processing units process the same program, the comparison unit compares resulting states from the processing of the same program by the two processing units for agreement.

44. The device as recited in claim **42**, wherein, in an operating mode corresponding to a safety mode, input data of the at least two processing units are compared with respect to one another for agreement.

45. The device as recited in claim **42**, wherein, in an operating mode corresponding to a safety mode, output data of the at least two processing units are compared with respect to one another for agreement.

46. The device as recited in claim **42**, wherein data to be distributed are relayed to a processing unit, and wherein the data to be distributed are extended by an error detection code prior to the relaying.

47. The device as recited in claim **42**, wherein one operating mode of the system is a performance mode, and wherein in the performance mode the data of both processing units are prioritized and at least one of sequentially received and relayed as a function of assigned priorities.

48. The device as recited in claim **42**, further comprising: a delay component which delays a preceding data by a clock pulse offset as a function of processing-unit clock pulse offset of the at least two processing units.

49. The device as recited in claim **42**, wherein data to be distributed are read from a memory and then distributed to the at least two processing units.

50. The device as recited in claim **42**, further comprising: at least one state machine controlling the distribution of the data.

51. The device as recited in claim **50**, wherein two state machines are provided for each processing unit.

52. The device as recited in claim **50**, wherein a synchronous state machine and an asynchronous state machine are provided.

53. A multi-processor system, comprising:

at least two processing units; and

device for performing data distribution from at least one data source, wherein the device includes a switchover unit which implements a switchover between at least two operating modes of the system;

wherein at least one of the data distribution and a selection of a data source depends on an operating mode of the system.

54. The system as recited in claim **53**, further comprising: a monitoring circuit external to the device for performing data distribution, wherein the monitoring circuit detects an error if an intended switchover between the at least two operating modes does not take place.

* * * * *