(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106407132 B
(45)授权公告日 2020.05.12

(21)申请号 201610830324.7

(22)申请日 2016.09.19

(65)同一申请的已公布的文献号
申请公布号 CN 106407132 A

(43)申请公布日 2017.02.15

(73)专利权人 复旦大学
地址 200433 上海市杨浦区邯郸路220号

(72)发明人 韩军 轩四中 袁腾跃 曾晓洋

(74)专利代理机构 上海正旦专利代理有限公司 31200
代理人 陆飞 陆尤

(51)Int.Cl.
G06F 12/0844(2016.01)

(56)对比文件
CN 101840390 A,2010.09.22,
US 2005060512 A1,2005.03.17,
CN 103440225 A,2013.12.11,
CN 102396171 A,2012.03.28,
WO 2016134380 A1,2016.08.25,
CN 102521201 A,2012.06.27,
罗康义.一种基于共享存储器的通信同步机制的实现方式.《船舶电子工程》.2004,89-92页.

审查员 贾超

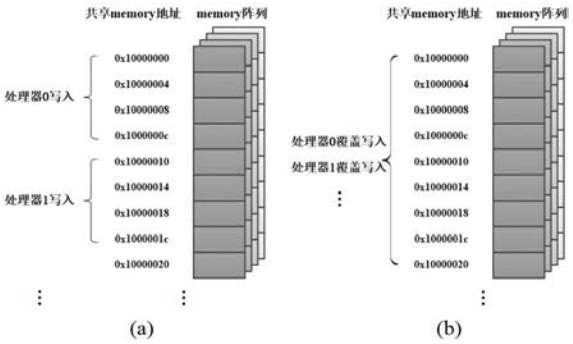
权利要求书1页 说明书4页 附图1页

(54)发明名称

一种基于共享存储器的数据通信同步方法

(57)摘要

本发明属于集成电路设计技术领域，具体为一种基于共享存储器的数据通信同步方法。在基于总线互连的多核SoC系统中，多个处理器之间的数据通信通常通过共享存储器来实现。本发明利用多核SoC实现并行GPDT算法，在矩阵运算中将各个处理器的计算结果利用共享存储器进行拼接或累加来构造完整结果。对于需要结果拼接的通信过程，根据每个处理器分配的工作量将共享地址空间合理地进行分割，使得每个处理器与分割之后的地址空间片段一一对应，将计算结果写入对应的地址空间；而对于需要结果累加的通信过程，则每个处理器覆盖写入同一块地址空间。为了避免数据发生冲突，本发明提供“检测标识-修改标识-访问存储-还原标识”的数据同步方法，这样大大降低了发生数据冲突的概率。



CN 106407132 B

1.一种基于共享存储器的数据通信同步方法,是基于双标识数据的,其特征在于,为每个处理器的每个共享数据块设计一个2 bit的标识位,用来表征该数据块对于其他处理器来说是否可读可写的情况;

设SoC系统共有N个处理器,则对于共享存储器中的每个共享数据块来说有2N bit的标识位;当某个处理器开始对共享存储器中某个数据块进行读写时,基本双标识的数据同步方法为:

读数据,具体步骤为:

(1)检测标识:读取标识并检测,如果数据不可读,等待;否则,进入步骤(2);

(2)修改标识:将读取标识暂存,并修改本核对应的标识为不可写;

(3)访问存储:将该数据块中的数据读取到本地存储器中;

(4)还原标识:将标识还原到检测标识之前的状态;

写数据,具体步骤为:

(1)检测标识:读取标识并检测,如果数据不可写,等待;否则,进入步骤(2);

(2)修改标志:将读取标识暂存,并修改本核对应的标识为不可读不可写;

(3)访问存储:将本地数据写入该核对应的数据块中;

(4)还原标识:将本核对应的标识改为不可读可写;

每次读取标识之后添加一个冗余写标识的操作,将标识全部设置为不可读不可写,即读取标识之后将标识锁住,等待标识的判断逻辑完成之后,再由本核对标识进行还原或者修改。

一种基于共享存储器的数据通信同步方法

技术领域

[0001] 本发明属于集成电路设计技术领域，具体为一种基于共享存储器的数据通信同步方法。

背景技术

[0002] 在共享存储的多核处理器中，一般会引入cache结构将共享存储空间的数据缓存到本地，利用其本身的结构特性来加速多核获取数据的过程。由于每个处理器看到的存储器视图都是通过本地cache得到的，因此对于同一个存储位置的数据来说，不同的处理器可能会获取到不同的数据值。在多核处理器设计中，必须引入cache一致性维护机制，确保每个处理器核对存储器中同一位置的读取操作总能获得最新写入的值。随着多核(多线程)之间数据交互模式的日趋复杂，能否提供高效的cache一致性维护机制，直接影响到多线程工作负载在多核上的运行效率及正确性。

[0003] 为了解决cache一致性问题，人们陆续提出了许多方案，其中公认比较有效并为商业多处理器系统所采用的有基于总线监听的MESI(或MOESI)协议和基于目录的协议等。

[0004] 通过引入cache来提高处理器访问数据的速度的代价对于对数据访问速度要求不高的系统来说是比较高昂的，因为需要解决维护cache一致性的问题。本发明需要解决的是在不引入cache的情况下，即处理器之间的数据通信仅通过共享存储器来实现，如何来确保多核之间数据通信的同步问题。基于本发明的多核SoC需要实现的是并行GPDT算法，每个处理器负责矩阵运算的一部分，然后将各个处理器的计算结果进行拼接或累加来构造完整结果。

发明内容

[0005] 本发明的目的在于提供一种在基于共享存储器的多核片上系统中解决多核之间数据通信的同步问题的方法。

[0006] 对于结果拼接的通信过程，根据每个处理器分配的工作量来将共享地址空间分割，使得每个处理器与分割后的地址空间片段一一对应，每个处理器只需将本核计算得到的数据块写入本核对应的地址空间即可；对于结果累加的通信过程，则无需将地址空间分割，每个处理器都从同一个起始地址开始覆盖写入数据即可。在多个核对共享存储器进行读写时，为了避免数据发生冲突，包括避免每个处理器在其他核写入数据之前读取或者在其他核读取过程中写入，本发明设计了一种双标识位的数据同步方法，即每个处理器按照“检测标识-修改标识-访问存储-还原标识”的机制来工作。

[0007] 为了对共享存储器的数据读写操作进行同步，本发明为每个处理器的每个共享数据块设计一个2 bit的标识位，用来表征该数据块对于其他处理器来说是否可读可写的情况；例如，标识位可设置如下：00，不可读不可写；01，不可读可写；10，可读不可写；11，可读可写。

[0008] 设SoC系统共有N个处理器，则对于共享存储器中的每个共享数据块来说有2N bit

的标识位;当某个处理器开始对共享存储器中某个数据块进行读写时,基于双标识的数据同步方法为:

[0009] 读数据,具体步骤为:

[0010] (1)检测标识:读取标识并检测,如果数据不可读,等待;否则,进入步骤(2);

[0011] (2)修改标识:将读取标识暂存,并修改本核对应的标识为不可写;

[0012] (3)访问存储:将该数据块中的数据读取到本地存储器中;

[0013] (4)还原标识:将标识还原到检测标识之前的状态。

[0014] 写数据,具体步骤为:

[0015] (1)检测标识:读取标识并检测,如果数据不可写,等待;否则,进入步骤(2);

[0016] (2)修改标志:将读取标识暂存,并修改本核对应的标识为不可读不可写;

[0017] (3)访问存储:将本地数据写入该核对应的数据块中;

[0018] (4)还原标识:将本核对应的标识改为不可读可写。

[0019] 冗余写:

[0020] 基于上述双标识数据同步规则来实现基于共享存储器的多核系统之间的通信,仍然有比较小的概率发生数据冲突,主要发生在标识的读写过程,因为每个核对标识的读取和修改之间存在标识的判断逻辑,会间隔一段时间,如果某个核需要写入共享存储器,在读取标识之后进行判断,检测到该数据块可读可写,但还未修改标识的时候,另外一个核需要读取共享存储器,对标识进行读取则检测到该数据块可读,那么会发生一个核写数据同时另一个核读数据的情况,很可能使得读取到的数据发生错误。为了避免这种数据冲突,本发明在每次读取标识之后添加一个冗余写标识的操作,将标识全部设置为不可读不可写,例如将全部标识置为00,即读取标识之后将标识锁住,等待标识的判断逻辑完成之后,再由本核对标识进行还原或者修改。这样大大降低了发生数据冲突的概率。

附图说明

[0021] 图1:共享存储器结构(a)结果拼接(b)结果累加。

[0022] 图2:算法中各函数表示的意义。

具体实施方式

[0023] 下面结合附图,对本发明作进一步的描述。

[0024] 本发明基于在多核SOC上实现并行GPDT算法。首先将每个需要共享的数据块在共享存储器中分配一块地址空间,对于结果拼接的通信过程,根据每个处理器分配到的工作量来将共享地址空间分割,使得每个处理器与分割之后的地址空间片段一一对应,每个处理器只需将本核计算得到的数据块写入本核对应的地址空间;对于结果累加的通信过程,则无需将地址空间分割,每个处理器从同一个起始地址开始覆盖写入,如图1所示。

[0025] 为每个处理器对应的共享数据块设计一个2bit的标识位,用来表征该数据块对于其他处理器来说是否可读可写的情况,标识位设置如表1所示:00,不可读不可写;01,不可读可写;10,可读不可写;11,可读可写。

[0026] 如果多核系统中有8个处理器,那么对于每个共享数据块来说共需要16bit的标识位来表征该数据块对于每个处理器是否可读可写的情况,该标识位也存储在共享存储器

中。

[0027] 当某个处理器开始对共享存储器中某个数据块进行读写时，按照如下“检测标识-修改标识-访问存储-还原标识”的数据同步机制来进行。

[0028] 1、读数据，具体步骤为：

[0029] (1)检测标识：读取标识并检测，如果有任何一个核对应的标识位00/01，则表示该数据块不可读，等待；否则进入步骤(2)；

[0030] (2)修改标识：如果该数据块可读，则先将读取的标识暂存，再将本核对应的标识改为10，表示该数据块正在被读取；

[0031] (3)访问存储：将该数据块中的数据读取到本地存储器中；

[0032] (4)还原标识：将标识还原到检测标识之前的状态。

[0033] 2、写数据，具体步骤为：

[0034] (1)检测标识：读取标识并检测，如果有任何一个核对应的标识位为00/10，则表示该数据块不可写，等待；否则进入步骤(2)；

[0035] (2)修改标识：如果该数据块可写，则先将读取的标识暂存，再将本核对应的的标识修改为00，表示该数据正在被写入；

[0036] (3)访问存储：将本地数据写入该核对应的数据块中；

[0037] (4)还原标识：将本核对应的标识改为01。

[0038] 注意，写入数据后并没有将标识还原为检测标识之前的状态，而是置为01.考虑到通信同步，是由主核来负责检测标识，当所有核的标识位都为01时，才由主核将负责将所有标识置为11。

[0039] 另外，由于处理器核对标识位的读取和修改之间存在判断逻辑，中间存在一段时间的间隔。如果某个核需要写入共享存储器，在读取标识之后，检测到该数据块可读可写，但还未修改标识的时候，另一个核需要读取共享存储器，就会检测该数据块可读，这时就会发生数据边写边读的情况，从而使另一个核读取到的数据发生错误，如图2所示。图2中各函数的意义如表2所示。为了避免这种情况发生，采取的措施是在每次读取标识位之后添加一个冗余写标识的操作，将全部标识置为00，对所有处理器不可读不可写，等待标识的判断逻辑完成之后，再由本核对标识进行还原或者修改，大大降低了数据发生冲突的概率。

[0040] 添加冗余写操作之后的完整的数据同步机制如下：

[0041] 1、读数据，具体步骤为：

[0042] (1)检测标识：读取标识，将共享存储器中所有标识置为00，检测读取到的标识，如果有任何一个核对应的标识位00/01，则表示该数据块不可读，还原标识，等待；否则进入2；

[0043] (2)修改标识：如果该数据块可读，则先将读取的标识暂存，再将本核对应的标识改为10，表示该数据块正在被读取；

[0044] (3)访问存储：将该数据块中的数据读取到本地存储器中；

[0045] (4)还原标识：将标识还原到检测标识之前的状态。

[0046] 2、写数据，具体步骤为：

[0047] (1)检测标识：读取标识，将共享存储器中所有标识置为00，检测读取到的标识，如果有任何一个核对应的标识位为00/10，则表示该数据块不可写，还原标识，等待；否则进入2；

[0048] (2)修改标识:如果该数据块可写,则先将读取的标识暂存,再将本核对应的的标识修改为00,表示该数据正在被写入;

[0049] (3)访问存储:将本地数据写入该核对应的数据块中;

[0050] (4)还原标识:将本核对应的标识改为01;

[0051] (5)还原标识(仅主核):读取并检测所有核的标识,当所有核的标识位都为01时,将所有标识还原为11;否则,等待。

[0052] 表1

[0053]

标识位	含义
00	不可读，不可写
01	不可读，可写
10	可读，不可写
11	可读，可写

[0054] 表2

[0055]

函数名	意义
readLabel()	读取标识（32bit 二进制数）
writingData()	将本地数据写入共享存储器对应的数据块
readingData()	从共享存储器中读取数据到本地存储器
writingLabel(data)	将总体标识置为 data（32bit 二进制数）
writingLocalLabel(lb)	将本核对应标识置为 lb（2bit 二进制数）
recoverLabel()	将标识还原到检测标识之前的状态

。

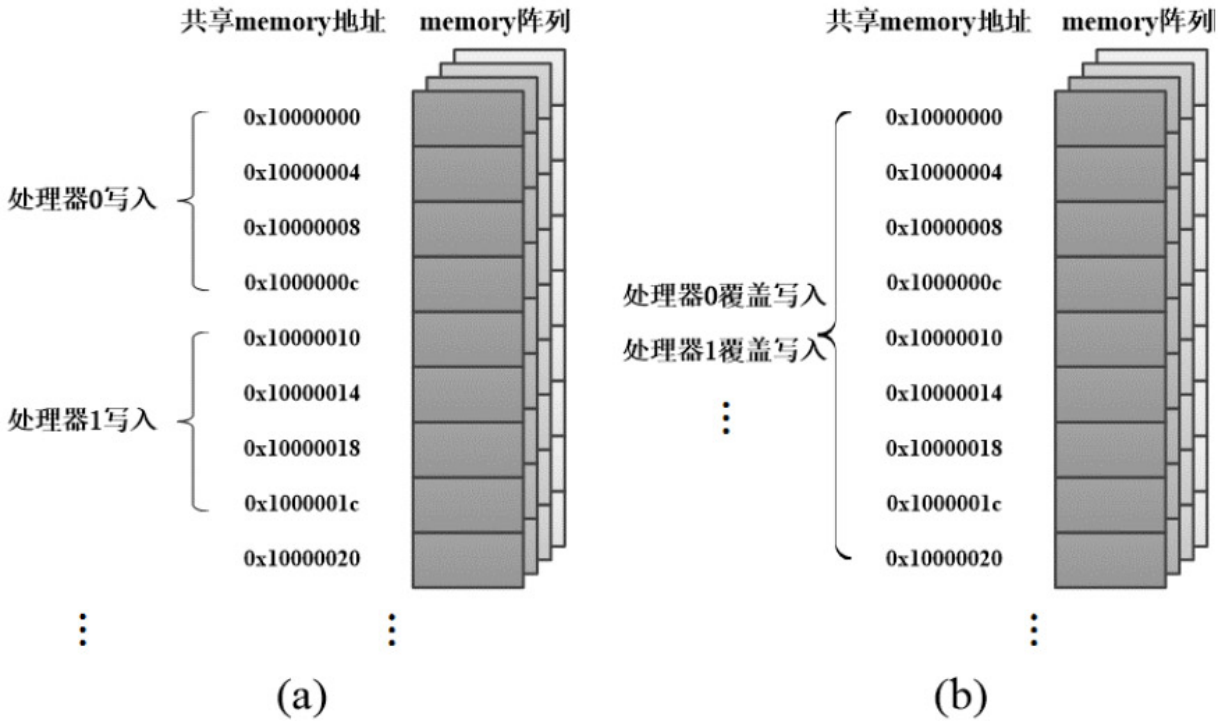

共享memory地址
memory阵列
处理器0写入
处理器1写入
0x10000000
0x10000004
0x10000008
0x1000000c
0x10000010
0x10000014
0x10000018
0x1000001c
0x10000020
(a)
共享memory地址
memory阵列
处理器0覆盖写入
处理器1覆盖写入
0x10000000
0x10000004
0x10000008
0x1000000c
0x10000010
0x10000014
0x10000018
0x1000001c
0x10000020
(b)

图1

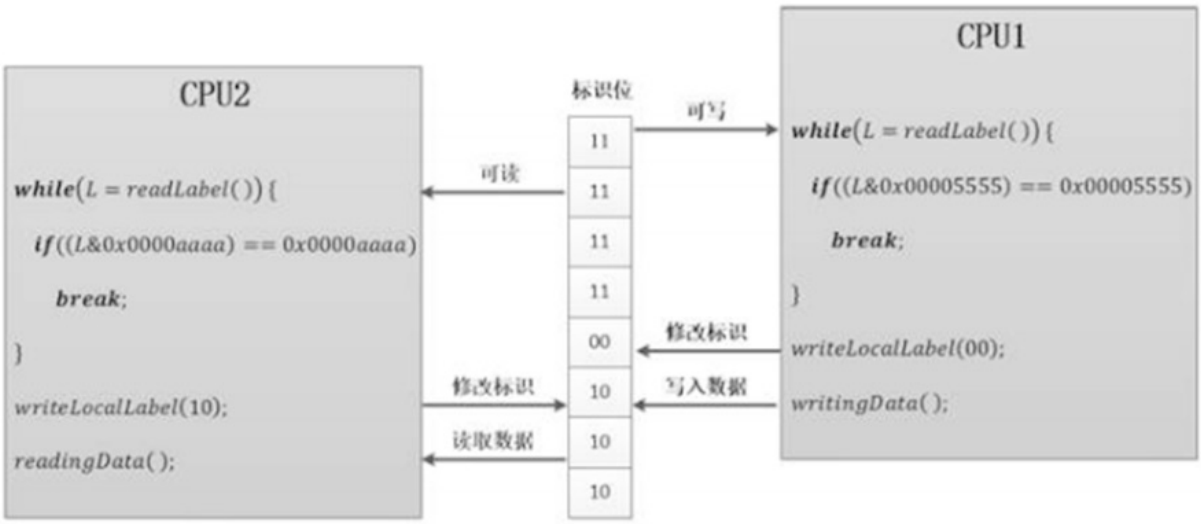

CPU2
while(L = readLabel()){
if((L&0x0000aaaa) == 0x0000aaaa)
break;
}
writeLocalLabel(10);
readingData();
CPU1
while(L = readLabel()){
if((L&0x00005555) == 0x00005555)
break;
}
writeLocalLabel(00);
writingData();
标识位
11
11
11
11
00
10
10
10
可写
可读
修改标识
写入数据
修改标识
读取数据

图2