(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 102521042 A
(43)申请公布日 2012.06.27

(21)申请号 201110422610.7

(22)申请日 2011.12.16

(71)申请人 中船重工(武汉)凌久电子有限责任公司
地址 430074 湖北省武汉市洪山区珞喻路718号

(72)发明人 舒红霞 王继红

(74)专利代理机构 湖北武汉永嘉专利代理有限公司 42102
代理人 王超

(51)Int.Cl.
G06F 9/48(2006.01)

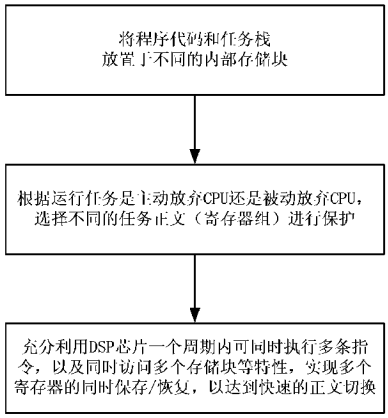
权利要求书 2 页 说明书 7 页 附图 3 页

(54)发明名称

基于哈佛结构DSP的快速正文切换方法

(57)摘要

本发明提供了一种基于哈佛结构DSP的快速正文切换方法,包括:S1)将程序代码和任务栈存放在DSP不同的内部存储块;S2)根据运行任务正文保护的种类,选择不同的寄存器组进行保护,并将寄存器组存放在相应的任务栈中;S3)当即将被调度运行的任务进行正文恢复时,根据该任务上次被调度出去时正文保护的种类,从相应的任务栈出栈,恢复曾保护的寄存器组的内容。本发明原理简单,易于实现,充分利用了DSP芯片的总线结构及处理特性,最高地发挥DSP性能,大大降低嵌入式实时操作系统任务切换的时间,提高嵌入式实时操作系统的实时性能。



CN 102521042 A

1. 基于哈佛结构 DSP 的快速正文切换方法，其特征在于，包括以下步骤：

S1) 将程序代码和任务栈存放在 DSP 不同的内部存储块；

S2) 根据运行任务正文保护的种类，选择不同的寄存器组进行保护，并将寄存器组存放在相应的任务栈中；

S3) 当即将被调度运行的任务进行正文恢复时，根据该任务上次被调度出去时正文保护的种类，从相应的任务栈出栈，恢复曾保护的寄存器组的内容。

2. 根据权利要求 1 所述的方法，其特征在于：所述 DSP 芯片，具有 3 个或 3 个以上内部存储块，任务栈包括任务 J 栈和任务 K 栈；或者，具有 2 个或 2 个以上内部存储块及一个任务栈。

3. 根据权利要求 2 所述的方法，其特征在于，步骤 S2 将寄存器组存放在相应的任务栈的方法包括：

a) 设需要保存的寄存器总数为 M，一个周期内的访问字长为 P；

b)M 除以 P 得到整数部分为 X，余数为 Y；

c)X 除以 2 得到整数部分为 W，余数为 Z；

d) 将 M 个寄存器中的 (W+Z)*P 个寄存器存放于任务 J 栈，剩余 W*P+Y 个寄存器存放于任务 K 栈。

4. 根据权利要求 2 所述的方法，其特征在于：步骤 S1 中，将程序代码、任务 J 栈、任务 K 栈存放在任意三个不相同的存储块中。

5. 根据权利要求 4 所述的方法，其特征在于，步骤 S2 选择需要保护的寄存器组的方法为：当运行任务主动放弃 CPU 使用权时，需保护系统保留寄存器组；当有更高优先级的任务到来而迫使运行任务被动放弃 CPU 使用权时，需要保护系统保留寄存器组和应用可访问寄存器组。

6. 根据权利要求 5 所述的方法，其特征在于，系统保留寄存器组包括：程序调用时系统保留的寄存器、堆栈指针寄存器以及指令跳转地址寄存器；应用可访问寄存器组包括程序调用时临时使用的寄存器。

7. 根据权利要求 5 所述的方法，其特征在于：如果即将被调度运行任务上次主动放弃 CPU 使用权，那么按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组的内容；如果即将被调度运行任务上次被动放弃 CPU 使用权，那么按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组和应用可访问寄存器组的内容。

8. 根据权利要求 5 所述的方法，其特征在于，所述 DSP 采用 ADSP-TS201S 型号。

9. 根据权利要求 8 所述的方法，其特征在于：步骤 S1 中，将程序代码、任务 J 栈、任务 K 栈分别存放在 M0、M2、M4、M6、M8、M10 中任意三个不相同的存储块中。

10. 根据权利要求 8 所述的方法，其特征在于：系统保留寄存器组包括 j16 ～ j27，k16 ～ k27，xr24 ～ xr31，yr24 ～ yr31，cjmp 和 reti；应用可访问寄存器组包含 xr0 ～ xr23，yr0 ～ yr23，j0 ～ j15，k0 ～ k15，j28 ～ j31，k28 ～ k31，jb0 ～ jb3，kb0 ～ kb3，jl0 ～ jl3，kl0 ～ kl3，xtr0 ～ xtr31，ytr0 ～ ytr31，xthr0 ～ xthr3，ythr0 ～ ythr3，xmr0 ～ xmr4，ymr0 ～ ymr4，xpr0 ～ xpr1，ypr0 ～ ypr1，xBFOTMP0 ～ xBFOTMP1，yBFOTMP0 ～ yBFOTMP1，xdab，ydab，xstat，ystat，lc0，lc1，sfreg；步骤 S2 将寄存器组存放在相应的任务栈的方法为：当运行任务主动放弃 CPU 使用权时，保存至任务 J 栈的寄存器包括：j16 ～ j27，xr24 ～

xr31,保存至任务 K 栈的寄存器主要包括 :k16 ～ k27, yr24 ～ yr31, cjmp 和 reti,当运行任务被动放弃 CPU 使用权时,保存至任务 J 栈的寄存器主要包括 :xr0 ～ xr31, j0 ～ j31, jb0 ～ jb3, jl0 ～ jl3, xtr0 ～ xtr31, xthr0 ～ xthr3, xdab, xpr0 ～ xpr1, xBFOTMP0 ～ xBFOTMP1, xmr0 ～ xmr4, xstat, lc0, sfreg,保存至任务 K 栈的寄存器包括 :yr0 ～ yr31, k0 ～ k31, kb0 ～ kb3, kl0 ～ kl3, ytr0 ～ ytr31, ythr0 ～ ythr3, ydab, ypr0 ～ ypr1, yBFOTMP0 ～ yBFOTMP1, ymr0 ～ ymr4, ystat, lc1, cjmp, reti。

基于哈佛结构 DSP 的快速正文切换方法

技术领域

[0001] 本发明涉及嵌入式实时操作系统中的任务调度领域，尤其是涉及一种具有哈佛结构的 DSP（数字信号处理器）上的实时操作系统快速正文切换方法。

背景技术

[0002] 高性能嵌入式并行数字信号处理机广泛应用于声纳信号处理、雷达信号处理、仿真训练、雷达成像等一切需要高速运算处理的领域。目前，针对并行数字信号处理系统，应用方式主要是前后台系统，没有操作系统支持。与传统的前后台系统应用方式相比，使用嵌入式实时操作系统开发并行数字信号处理应用的优点主要有：

[0003] 第一，传统的前后台系统应用方式对并行数字信号处理应用开发人员的要求极高，开发人员不仅需要掌握数字信号处理的核心算法，还需要对数字信号处理系统的底层结构十分清楚。操作系统提供了与硬件系统无关的抽象层，屏蔽了硬件环境的相关细节，开发人员可以不必关注底层硬件系统，把精力放在应用算法上，这样能大大地提高应用开发速度。

[0004] 第二，早期的并行数字信号处理应用很简单，软件的编程也较为简单。随着系统越来越复杂庞大，并行数字信号处理应用也越来越复杂，应用程序需要划分成多个重要性不同的任务，并且，在各任务间优化地分配 CPU 时间和系统资源的同时，还需要保证实时性。并行数字信号处理应用开发人员使用操作系统进行开发，可以按处理流程自然地划分，而不会像传统的前后台系统那样生硬地将算法分割映射。

[0005] 第三，当数字信号处理器芯片升级，数字信号处理系统更换时，使用传统的前后台系统方式开发的应用需要根据新的数字信号处理芯片及数字信号处理系统特点重新开发，导致了同一个应用的重复开发，浪费了大量时间。而操作系统封装了与硬件相关的代码，硬件系统对于开发人员透明，因此，使用操作系统开发的并行数字信号处理应用具有相当好的移植性。

[0006] 第四，当并行数字信号处理系统中 I/O（输入 / 输出）频繁，需要响应的实时事件过多，传统的前后台系统的实时性将会变得很差，而操作系统可以对 I/O 进行有效地管理，使得 I/O 不再成为系统瓶颈。

[0007] 第五，针对并行数字信号处理系统，传统的前后台系统应用方式下不同处理器节点之间的通讯相当复杂，编程十分困难，而操作系统提供的消息传递功能，屏蔽了硬件系统互联及消息的物理传输细节，应用开发人员只需要关注消息的发送任务和接收任务，至于消息传递过程中涉及的发送节点、转发节点和目的节点之间的物理链路完全不必关心，这种编程方式十分简单，移植起来也相当方便。

[0008] 综上所述，针对并行数字信号处理系统，设计并实现一个高性能嵌入式实时操作系统成为一种必然需求。任务调度是操作系统的核心和灵魂，决定了操作系统的实时性能。当操作系统内核调度运行其它任务时，需要将当前正在运行任务的正文保存，同时恢复即将调度的任务正文，任务正文的保存和恢复时间决定了任务正文切换时间，将直接影响调

度时间。DSP 芯片中大多采用哈佛结构，即一种将程序指令存储器和数据存储器分开的存储器结构，它采用独立的地址总线和数据总线完成程序和数据的访问。因此，充分利用 DSP 芯片这一特性，设计并实现一个快速的正文切换方法十分必要。

[0009] 到目前为止，还没有相关研究涉及到利用 DSP 哈佛结构这一特性，对嵌入式实时操作系统实现快速正文切换的方法。

发明内容

[0010] 本发明目的在于提供一种具有哈佛结构的 DSP 上的实时操作系统快速正文切换方法。本发明尽可能地利用 DSP 的哈佛结构及多条内部总线连接不同的存储块这一特点，通过在同一周期内，访问不同的存储块，以及多条指令并行执行，实现任务正文的快速保存和恢复。

[0011] 本发明采用的技术方案如下：基于哈佛结构 DSP 的快速正文切换方法，包括以下步骤：

[0012] S1) 将程序代码和任务栈存放在 DSP 不同的内部存储块；

[0013] S2) 根据运行任务正文保护的种类，选择不同的寄存器组进行保护，并将寄存器组存放在相应的任务栈中；

[0014] S3) 当即将被调度运行的任务进行正文恢复时，根据该任务上次被调度出去时正文保护的种类，从相应的任务栈出栈，恢复曾保护的寄存器组的内容。

[0015] 所述的方法，所述 DSP 芯片，具有 3 个或 3 个以上内部存储块，任务栈包括任务 J 栈和任务 K 栈；或者，具有 2 个或 2 个以上内部存储块及一个任务栈。

[0016] 所述的方法，步骤 S2 将寄存器组存放在相应的任务栈的方法包括：

[0017] a) 设需要保存的寄存器总数为 M，一个周期内的访问字长为 P；

[0018] b)M 除以 P 得到整数部分为 X，余数为 Y；

[0019] c)X 除以 2 得到整数部分为 W，余数为 Z；

[0020] d) 将 M 个寄存器中的 (W+Z)*P 个寄存器存放于任务 J 栈，剩余 W*P+Y 个寄存器存放于任务 K 栈。

[0021] 所述的方法，步骤 S1 中，将程序代码、任务 J 栈、任务 K 栈存放在任意三个不相同的存储块中。

[0022] 所述的方法，步骤 S2 选择需要保护的寄存器组的方法为：当运行任务主动放弃 CPU 使用权时，需保护系统保留寄存器组；当有更高优先级的任务到来而迫使运行任务被动放弃 CPU 使用权时，需要保护系统保留寄存器组和应用可访问寄存器组。

[0023] 所述的方法，系统保留寄存器组包括：程序调用时系统保留的寄存器、堆栈指针寄存器以及指令跳转地址寄存器；应用可访问寄存器组包括程序调用时临时使用的寄存器。

[0024] 所述的方法，如果即将被调度运行任务上次主动放弃 CPU 使用权，那么按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组的内容；如果即将被调度运行任务上次被动放弃 CPU 使用权，那么按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组和应用可访问寄存器组的内容。

[0025] 所述的方法，所述 DSP 采用 ADSP-TS201S 型号，

[0026] 所述的方法，步骤 S1 中，将程序代码、任务 J 栈、任务 K 栈分别存放在 M0、M2、M4、

M6、M8、M10 中任意三个不相同的存储块中。

[0027] 所述的方法，系统保留寄存器组包括 j16 ～ j27，k16 ～ k27，xr24 ～ xr31，yr24 ～ yr31，cjmp 和 reti ；应用可访问寄存器组包含 xr0 ～ xr23，yr0 ～ yr23，j0 ～ j15，k0 ～ k15，j28 ～ j31，k28 ～ k31，jb0 ～ jb3，kb0 ～ kb3，jl0 ～ jl3，kl0 ～ kl3，xtr0 ～ xtr31，ytr0 ～ ytr31，xthr0 ～ xthr3，ythr0 ～ ythr3，xmr0 ～ xmr4，ymr0 ～ ymr4，xpr0 ～ xpr1，ypr0 ～ ypr1，xBFOTMP0 ～ xBFOTMP1，yBFOTMP0 ～ yBFOTMP1，xdab，ydab，xstat，ystat，lc0，lc1，sfreg ；步骤 S2 将寄存器组存放在相应的任务栈的方法为 ：当运行任务主动放弃 CPU 使用权时，保存至任务 J 栈的寄存器包括 ：j16 ～ j27，xr24 ～ xr31，保存至任务 K 栈的寄存器主要包括 ：k16 ～ k27，yr24 ～ yr31，cjmp 和 reti，当运行任务被动放弃 CPU 使用权时，保存至任务 J 栈的寄存器主要包括 ：xr0 ～ xr31，j0 ～ j31，jb0 ～ jb3，jl0 ～ jl3，xtr0 ～ xtr31，xthr0 ～ xthr3，xdab，xpr0 ～ xpr1，xBFOTMP0 ～ xBFOTMP1，xmr0 ～ xmr4，xstat，lc0，sfreg，保存至任务 K 栈的寄存器包括 ：yr0 ～ yr31，k0 ～ k31，kb0 ～ kb3，kl0 ～ kl3，ytr0 ～ ytr31，ythr0 ～ ythr3，ydab，ypr0 ～ ypr1，yBFOTMP0 ～ yBFOTMP1，ymr0 ～ ymr4，ystat，lc1，cjmp，reti。

[0028] 本发明具有的有益效果是 ：原理简单，易于实现，充分利用了 DSP 芯片的总线结构及处理特性，最高地发挥 DSP 性能，大大降低嵌入式实时操作系统任务切换的时间，提高嵌入式实时操作系统的实时性能。本发明特别适合应用于一种具有哈佛结构的 DSP 上的实时操作系统任务调度领域。

附图说明

[0029] 图 1 是本发明单栈结构存储块划分示意图。

[0030] 图 2 是本发明双栈结构存储块划分示意图。

[0031] 图 3 是本发明任务正文分类示意图。

[0032] 图 4 是本发明单栈结构任务正文保护示意图。

[0033] 图 5 是本发明双栈结构任务正文保护示意图。

[0034] 图 6 是本发明的整个流程框图。

具体实施方式

[0035] 下面阐述本发明的技术内容和工作原理。

[0036] (1) 内部存储块划分

[0037] 对于采用单一堆栈结构的 DSP 芯片，将程序代码和任务栈存放在不同的内部存储块中，如图 1 所示。

[0038] 对于采用双堆栈结构（称其中一个堆栈为任务 J 栈，另一个堆栈为任务 K 栈）的 DSP 芯片，将程序代码、任务 J 栈和任务 K 栈存放于不同的内部存储块，如图 2 所示。

[0039] (2) 任务正文分类

[0040] 任务正文实际是指 CPU 寄存器内容。当操作系统内核决定运行其它任务时，需要将当前正在运行任务的正文保存至正文保护区中，正文保护区位于任务的运行栈。当完成当前任务的正文保护后，立即恢复即将调度的任务正文，即重新载入 CPU 寄存器，同时运行该任务。

[0041] 任务正文的保存和恢复增加了任务切换的时间，CPU 寄存器越多，所花费的时间越长。也就是说，被保护 CPU 寄存器的数目决定了任务切换的时间。并不是每次任务切换都需要保护所有的 CPU 寄存器。当运行任务主动放弃 CPU 使用权时，只需要保护系统保留寄存器组，当有更高优先级的任务到来而迫使运行任务被动放弃 CPU 使用权时，需要保护系统保留寄存器组和应用可访问寄存器组，如图 3 所示。

[0042] (3) 任务正文保存 / 恢复

[0043] 如果 DSP 芯片采用单一堆栈结构，那么保存 / 恢复任务正文时，寄存器组入栈 / 出栈操作在同一个任务栈中进行，如图 4 所示。并且，充分利用 DSP 芯片的一个周期内可同时执行多条指令，以及支持双字、四字访问等特性，实现多个寄存器的同时保存 / 恢复，以达到最快速的任务正文保存 / 恢复。

[0044] 如果 DSP 芯片采用双堆栈结构，那么保存 / 恢复任务正文时，寄存器组入栈 / 出栈操作在任务 J 栈和任务 K 栈中同时进行，如图 5 所示。这时，可以将寄存器组分成两部分，按照一定的顺序，将其中一部分存入任务 J 栈，另一部分存入任务 K 栈；出栈时，从任务 J 栈和任务 K 栈上，按照先进后出的顺序依次载入 CPU 寄存器组。并且，充分利用 DSP 芯片的一个周期内可同时执行多条指令，支持双字、四字访问，以及可同时访问不同的存储块等特性，实现任务 J 栈和任务 K 栈上的寄存器组的同时保存 / 恢复，以达到最快速的任务正文保存 / 恢复。

[0045] 下面结合附图进一步详述本发明。本实施方式是本发明的方法在一个并行 DSP 系统中的实施。这种并行 DSP 系统采用的 DSP 芯片具有如下特点：

[0046] (1) 具有 3 个或 3 个以上内部存储块，将这些存储块分别标记为：存储块 1、存储块 2、…、存储块 n(n ≥ 3)。

[0047] (2) 具有双堆栈结构，称其中一个堆栈为任务 J 栈，另一个堆栈为任务 K 栈。

[0048] (3) 具有 3 条总线，其中，1 条为指令传输总线，1 条为 J 数据传输总线，1 条为 K 数据传输总线。

[0049] (4) 同一个周期内，可访问 4 条 32 位指令、8 个 32 位数据字。

[0050] (5) 同一个周期内，支持四字的数据、指令及 I/O 访问。

[0051] 下面详细说明本实施方式的具体实现：

[0052] (1) 内部存储块划分

[0053] 将程序代码、任务 J 栈、任务 K 栈存放于存储块 0、存储块 1、…、存储块 n 中任意三个存储块，并且，这三个存储块互不相同。例如：如图 3 所示，将程序代码存放于存储块 i，任务 J 栈存放于存储块 j，任务 K 栈存放于存储块 k，其中 i、j、k 必须满足下列条件：1 ≤ i，j，k ≤ n，且 i ≠ j ≠ k。

[0054] 这样，在同一个周期内，DSP 处理器可以并行访问 i、j、k 这 3 个存储块，实现 1 次取指令、2 次访问数据操作，完成任务正文的高速存取。

[0055] (2) 任务正文分类

[0056] 将本实施方式中采用的 DSP 芯片的寄存器分成下列 4 类：

[0057] a) 程序调用时系统保留的寄存器 (reserved 类型寄存器)，如果在调用的子程序中使用了这类寄存器，那么当子程序返回时必须恢复其值。

[0058] b) 程序调用时临时使用的寄存器 (scratch 类型寄存器)，如果在调用的子程序中

使用了这类寄存器，那么当子程序返回时不需要恢复其值。

[0059] c) 堆栈指针寄存器，即存放堆栈内存储单元偏移量的寄存器。

[0060] d) 指令跳转地址寄存器，即跳转指令存放跳转地址的寄存器。

[0061] 将上述四类寄存器分成系统保留寄存器组和应用可访问寄存器组。其中，系统保留寄存器组是由程序调用时系统保留的寄存器 (reserved 类型的寄存器)、堆栈指针寄存器以及指令跳转地址寄存器组成的集合。应用可访问寄存器组是由程序调用时临时使用的寄存器 (scratch 类型的寄存器) 组成的集合。

[0062] 当运行任务主动放弃 CPU 使用权时，只需保护系统保留寄存器组，那么，任务正文由系统保留寄存器组构成。

[0063] 当有更高优先级的任务到来而迫使运行任务被动放弃 CPU 使用权时，需要保护系统保留寄存器组和应用可访问寄存器组，那么，任务正文由系统保留寄存器组和应用可访问寄存器组构成。

[0064] (3) 任务正文保存 / 恢复

[0065] 本实施方式采用的 DSP 芯片具有双堆栈结构及四字传输的特性，那么，当保存任务正文时，将需要保存的所有寄存器按照下列方法进行划分 :

[0066] a) 设需要保存的寄存器总数为 M。

[0067] b)M 除以 4 得到整数部分为 X，余数为 Y。

[0068] c)X 除以 2 得到整数部分为 W，余数为 Z。

[0069] d) 将 M 个寄存器中的 (W+Z)*4 个寄存器存放于任务 J 栈，剩余 W*4+Y 个寄存器存放于任务 K 栈。

[0070] 当运行任务主动放弃 CPU 使用权时，只需要保护系统保留寄存器组，按照上述划分方法，将系统保留寄存器组中的所有寄存器依次保存至任务 J 栈和任务 K 栈。

[0071] 当运行任务被动放弃 CPU 使用权时，需要保护系统保留寄存器组和应用可访问寄存器组，按照上述划分方法，将系统保留寄存器组和应用可访问寄存器组中的所有寄存器依次保存至任务 J 栈和任务 K 栈。

[0072] 当运行任务正文保存完成，需要恢复即将调度任务的正文。这时，根据即将调度任务上次被调度出去时的方式进行恢复，如果上次是主动放弃 CPU 使用权，那么仅按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组的内容 ;如果上次是被动放弃 CPU 使用权，那么按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组和应用可访问寄存器组的内容。

[0073] 当保存 / 恢复寄存器组内容时，在同一个周期内，采用四字寄存器内容同时存取，以及并行执行访问任务 J 栈和任务 K 栈的指令，实现快速的正文保存 / 恢复。具体实现如下 :

[0074] 保存恢复寄存器组内容，可以采用如下宏定义 :

[0075] #define PUSH(sp,reg)Q[sp+ = -0x4] = reg// 表示四字寄存器内容同时存储

[0076] #define POP(sp,reg)reg = Q[sp+ = 0x4]// 表示四字寄存器内容同时恢复

[0077] 保存寄存器组内容，可以采用下列伪代码 :

[0078] PUSH(jsp,reg),PUSH(ksp,reg) ;// 表示两条入栈指令并行执行

[0079] 恢复寄存器组内容，可以采用下列伪代码 :

[0080] POP(jsp,reg),POP(ksp,reg);// 表示两条出栈指令并行执行

[0081] 上述宏定义及伪代码中,sp 表示栈顶指针,reg 表示四字寄存器,PUSH 为入栈,POP 为出栈,Q 为四字访问,jsp 表示任务 J 栈栈顶指针,ksp 表示任务 K 栈栈顶指针。

[0082] 这种任务正文保存 / 恢复方式,充分利用了本实施方式采用的 DSP 芯片在一个周期内可访问 4 条 32 位指令和 8 个 32 位数据字,同时支持四字的数据、指令及 I/O 访问等特性,实现任务 J 栈和任务 K 栈上的寄存器的同时保存 / 恢复,以达到最快速的任务正文保存 / 恢复。

[0083] 实施例 :

[0084] 本实施例是本发明的方法在一个并行 DSP 系统中实施的实例。该并行 DSP 系统采用 ADSP-TS201S 处理器 (简称 TS201)。本实施例涉及的 TS201 处理器采用双堆栈结构,且具有 M0、M2、M4、M6、M8 和 M10 六个存储块,在同一个周期内,可进行四字的数据、指令及 I/O 访问。

[0085] 下面详细说明本实施例的具体实现 :

[0086] (1) 内部存储块划分

[0087] 将程序代码、任务 J 栈、任务 K 栈分别存放在 M0、M2、M4、M6、M8、M10 中任意三个不相同的存储块中,例如 :将程序代码存放于存储块 M0,将任务 J 栈存放于存储块 M6,将任务 K 栈存放于存储块 M8。这样,在同一个周期内,TS201 处理器可并行访问这 3 个存储块,实现 1 次取指令、2 次访问数据操作,完成任务正文的高速存取。

[0088] (2) 任务正文划分

[0089] 对于 TS201 处理器,系统保留寄存器组主要包括 :j16 ～ j27,k16 ～ k27,xr24 ～ xr31,yr24 ～ yr31,cjmp 和 reti。应用可访问寄存器组主要包括 :xr0 ～ xr23,yr0 ～ yr23,j0 ～ j15,k0 ～ k15,j28 ～ j31,k28 ～ k31,jb0 ～ jb3,kb0 ～ kb3,jl0 ～ jl3,kl0 ～ kl3,xtr0 ～ xtr31,ytr0 ～ ytr31,xthr0 ～ xthr3,ythr0 ～ ythr3,xmr0 ～ xmr4,ymr0 ～ ymr4,xpr0 ～ xpr1,ypr0 ～ ypr1,xBFOTMP0 ～ xBFOTMP1,yBFOTMP0 ～ yBFOTMP1,xdab,ydab,xstat,ystat,lc0,lc1,sfreg。

[0090] 当运行任务主动放弃 CPU 使用权时,只需保护系统保留寄存器组,当有更高优先级的任务到来而迫使运行任务放弃 CPU 使用权时,需要保护系统保留寄存器组和应用可访问寄存器组。

[0091] (3) 任务正文保存 / 恢复

[0092] 当运行任务主动放弃 CPU 使用权时,只需要保护系统保留寄存器组。TS201 处理器采用双堆栈结构,那么保存任务正文时,保存至任务 J 栈的寄存器包括 :j16 ～ j27,xr24 ～ xr31,保存至任务 K 栈的寄存器主要包括 :k16 ～ k27,yr24 ～ yr31,cjmp 和 reti。

[0093] 当运行任务被动放弃 CPU 使用权时,需要保护系统保留寄存器组和应用可访问寄存器组,保存至任务 J 栈的寄存器主要包括 :xr0 ～ xr31,j0 ～ j31,jb0 ～ jb3,jl0 ～ jl3,xtr0 ～ xtr31,xthr0 ～ xthr3,xdab,xpr0 ～ xpr1,xBFOTMP0 ～ xBFOTMP1,xmr0 ～ xmr4,xstat,lc0,sfreg。保存至任务 K 栈上的寄存器包括 :yr0 ～ yr31,k0 ～ k31,kb0 ～ kb3,kl0 ～ kl3,ytr0 ～ ytr31,ythr0 ～ ythr3,ydab,ypr0 ～ ypr1,yBFOTMP0 ～ yBFOTMP1,ymr0 ～ ymr4,ystat,lc1,cjmp,reti。

[0094] 当运行任务正文保存完成,需要恢复即将调度的任务正文。这时根据即将调度任

务上次被调度出去时的方式，即如果上次是主动放弃 CPU 使用权，那么仅按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组的内容；如果上次是被动放弃 CPU 使用权，那么按照先进后出的顺序依次从任务 J 栈和任务 K 栈上恢复系统保留寄存器组和应用可访问寄存器组的内容。

[0095] 实测性能指标：

[0096] 在处理器时钟为 600MHZ 的 ADSP-TS201S 上，采用本实施例后，嵌入式实时操作系统任务切换时间为 442ns，中断延迟时间为 140ns。因此，本发明可以获得很短的嵌入式实时操作系统任务切换时间，大大提高嵌入式实时操作系统的实时性能。

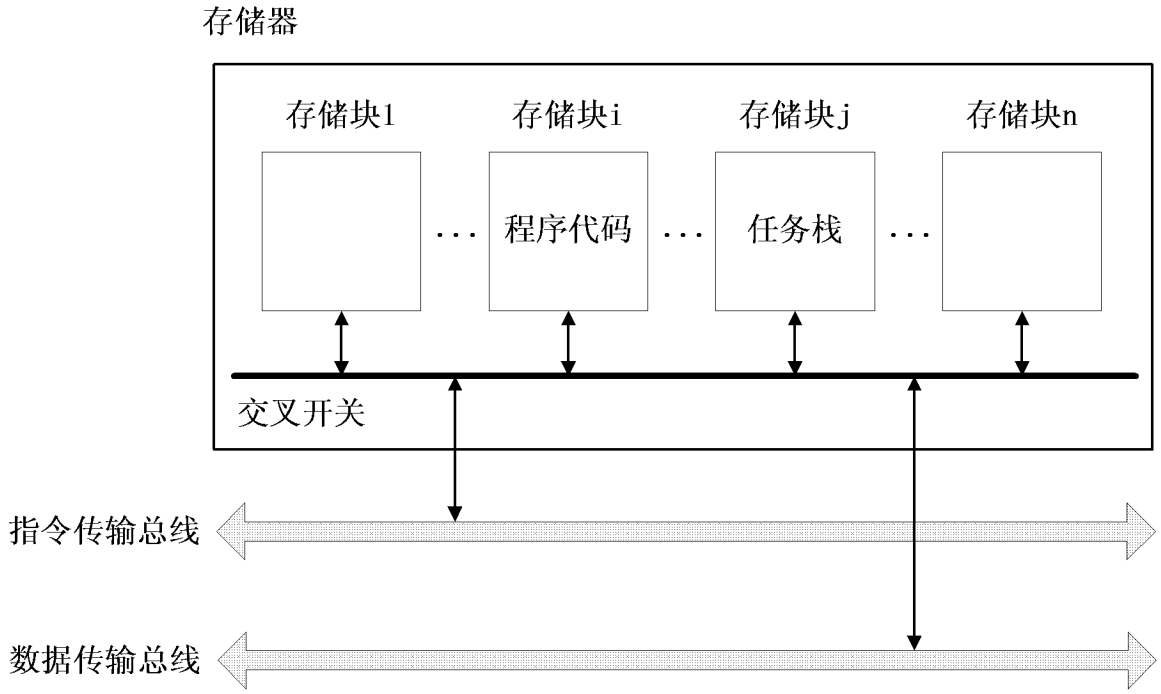

存储器
存储块1
存储块i
存储块j
存储块n
程序代码
任务栈
交叉开关
指令传输总线
数据传输总线

图 1

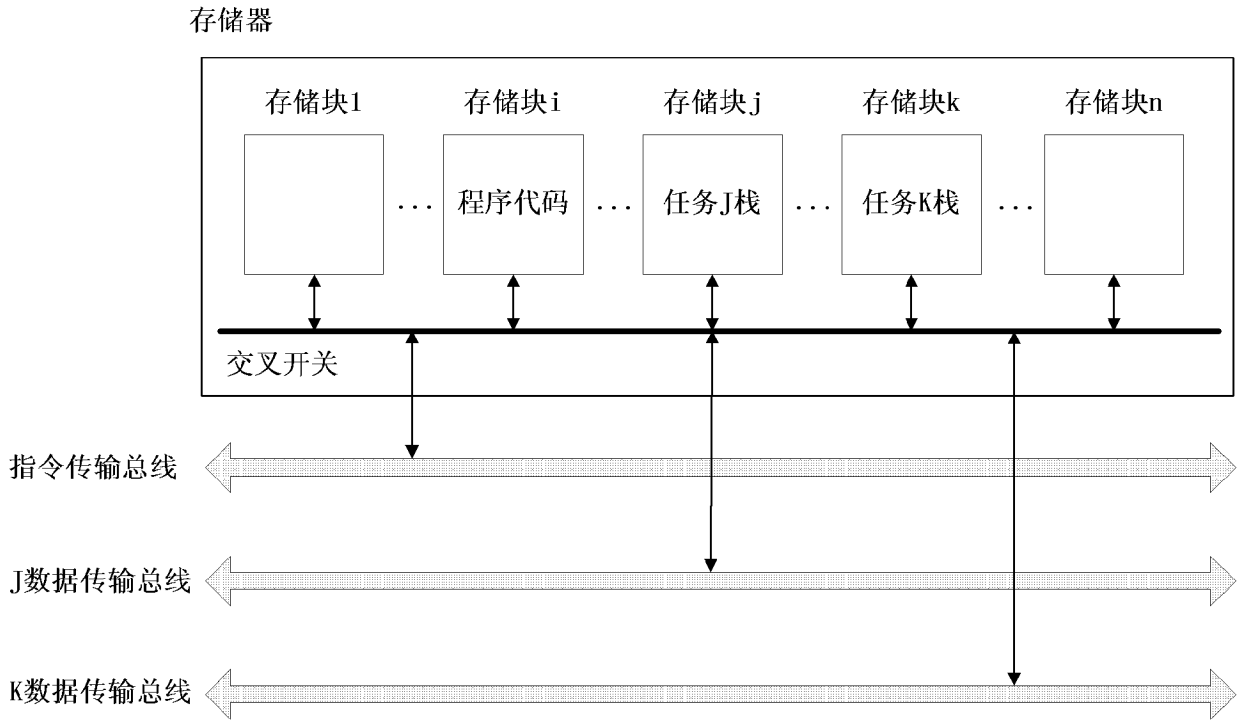

存储器
存储块1
存储块i
存储块j
存储块k
存储块n
程序代码
任务J栈
任务K栈
交叉开关
指令传输总线
J数据传输总线
K数据传输总线

图 2

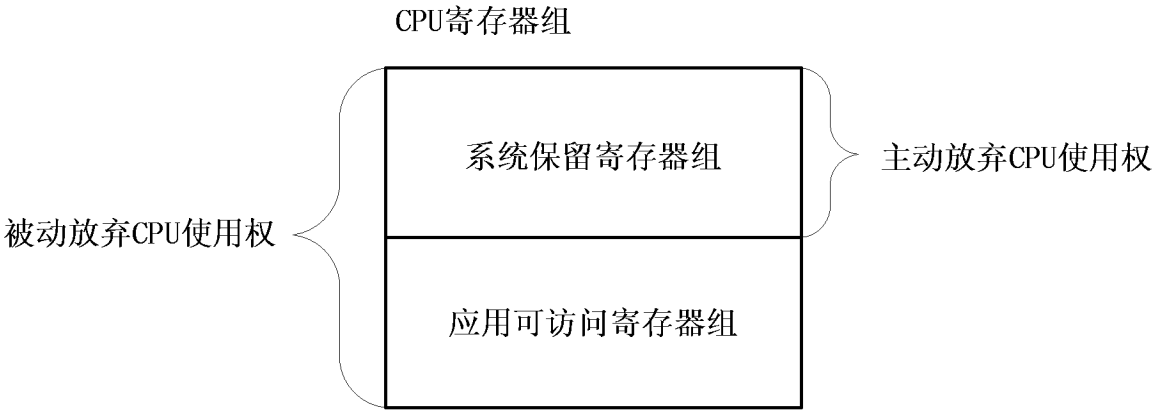

CPU寄存器组
系统保留寄存器组
应用可访问寄存器组
主动放弃CPU使用权
被动放弃CPU使用权

图 3

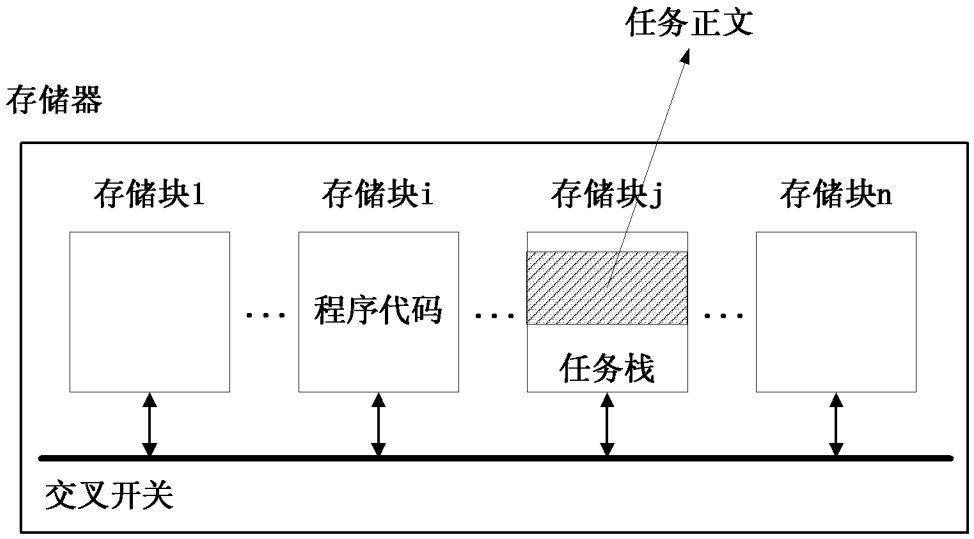

任务正文
存储器
存储块1
存储块i
存储块j
存储块n
程序代码
任务栈
交叉开关

图 4

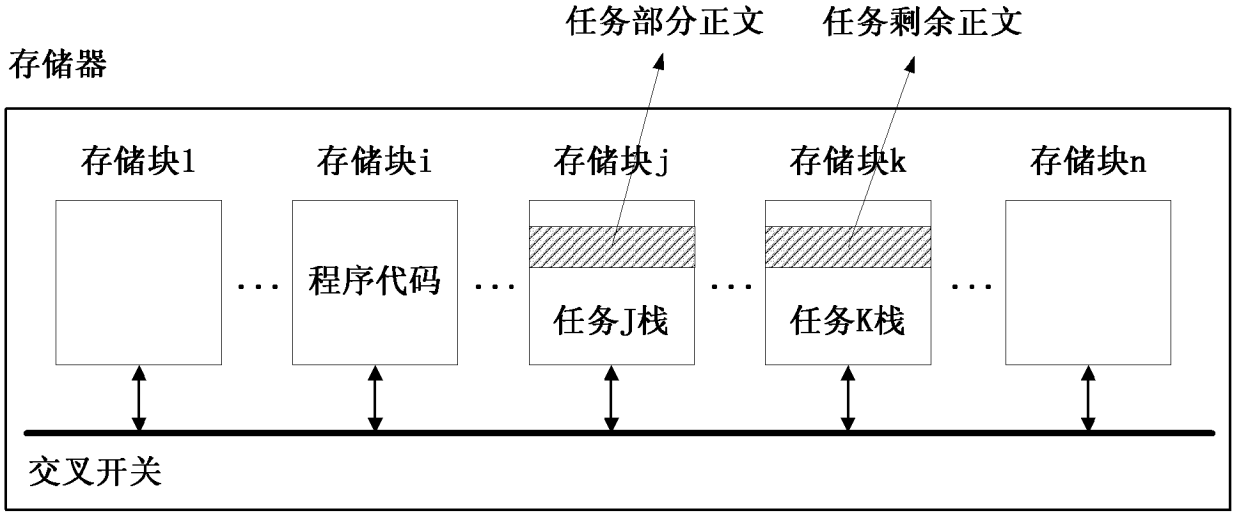

任务部分正文
任务剩余正文
存储器
存储块1
存储块i
存储块j
存储块k
存储块n
程序代码
任务J栈
任务K栈
交叉开关

图 5

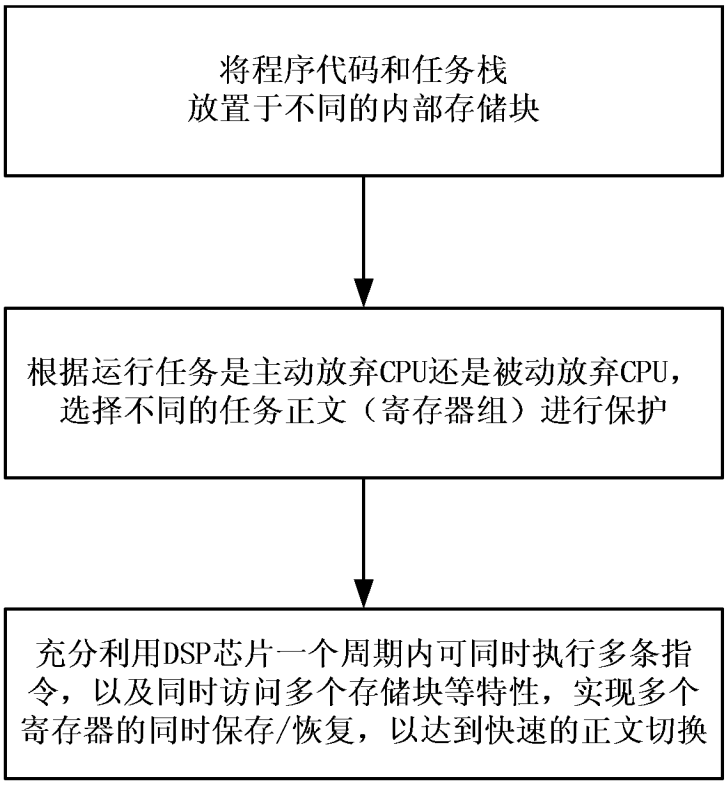

将程序代码和任务栈
放置于不同的内部存储块
根据运行任务是主动放弃CPU还是被动放弃CPU，
选择不同的任务正文（寄存器组）进行保护
充分利用DSP芯片一个周期内可同时执行多条指令，以及同时访问多个存储块等特性，实现多个寄存器的同时保存/恢复，以达到快速的正文切换

图 6