文章编号: 1005-3026(2006)09-0968-04

一种嵌入式硬件多线程处理器的研究

尹震宇, 赵 海, 张文波, 王小英
(东北大学 信息科学与工程学院, 辽宁 沈阳 110004)

摘 要: 提出了一种基于同时多线程技术的硬件多线程处理器设计。通过处理器内部的硬件机制来完成对多线程的调度管理, 实现基于硬件的时间片轮询多线程调度机制。最大程度地减少操作系统中关于线程调度的开销, 提高处理器执行多用户线程时的整体效率, 简化了用户在多线程条件下的编程复杂度, 增强了多线程运行环境下处理器对线程的保护。

关 键 词: 多线程处理器; 多线程处理; FPGA; 嵌入式系统; 处理器设计
中图分类号: TP 302.7; TP 316.2 文献标识码: A

在目前的嵌入式系统中, 多线程调度的实现多是建立在软件层面上的, 即通过操作系统来完成对线程调度的管理, 该方法存在以下不足: 首先, 系统在进行时间片轮询切换操作时将消耗大量的时间, 降低处理器执行用户线程的效率。其次, 大多数的嵌入式处理器在硬件上基本没有设计多线程执行环境下的硬件保护机制, 因而在运行多线程程序时可能会存在安全漏洞。此外, 用户对于多线程编程开发, 还存在着需要依赖系统函数、编程实现较复杂的问题。因而本文提出一种采用同时多线程SMT(simultaneous multi-threading)技术实现的硬件多线程处理器设计。通过处理器内部的硬件机制来完成对多线程的调度管理, 最大程度地减少操作系统中关于线程调度的开销, 提高处理器执行用户线程的整体效率, 简化了用户在多线程条件下的编程复杂度, 增强了在多线程运行环境下对各线程的保护。

1 相关工作

随着处理器设计技术和芯片集成度的提高, 在一个处理器上实现多个逻辑处理器, 进而实现硬件多线程的并发运行已经变得可行[1,2]。对于嵌入式系统来说, 在设计上需要考虑设计复杂度和成本[3~5], 因而在本文中, 提出一种"单执行内核的同时多线程"设计。即通过在传统处理核心基础上外扩一个由硬件逻辑控制的多线程调度管理电路, 来实现硬件层面上多个线程的调度操作。处理器的线程调度单元将外存中的多个线程的指令代码按顺序取入待执行线程队列, 并且按时间片顺序将待执行队列中的多个线程的指令代码轮流送入执行单元中执行。在这样的处理器结构中, 每一个硬件控制的线程轮流在自己的时间片内使用处理器的资源。而每一个由处理器硬件控制的线程, 在处理器的执行单元中运行时所看到的处理器是一个完整的单任务处理器, 即在某一个特定时间内只有一个线程使用处理器的各种资源。

采用此种结构, 所设计的处理器中只存在一个物理处理器执行核, 降低了设计的复杂度, 而所实现的处理器的规模也比同等级的单线程处理器增加不多, 因此, 采用SMT技术可以实现在尽量少的操作系统软件参与下, 以较低的硬件成本在单处理器芯片系统上实现硬件多线程操作。

2 硬件多线程处理器核的设计

硬件多线程处理器的硬件调度控制部分的设计主要包括: 指令节拍的状态机的设计、硬件多线程处理器的结构设计及对寄存器存储器资源保护的设计。

2.1 指令节拍的状态机设计

在本设计中采用了摩尔有限状态机[6,7], 将一条指令的执行分解为取指、译码、执行、访存及回写五个部分, 并具体将这五个部分细化为状态机中的IFdelay(instruction fetch delay)、IF(instruction fetch)、DI(decode instruction)、EX

收稿日期: 2005-12-07
基金项目: 国家高技术研究发展计划项目(2001AA415320)。
作者简介: 尹震宇(1979—), 男, 辽宁沈阳人, 东北大学博士研究生; 赵 海(1959—), 男, 辽宁沈阳人, 东北大学教授, 博士生导师。

(execute)、MEMdelay(memory access delay)、MEM(memory access)和 WB(register write back)七个状态。

2.2　硬件多线程的实现机制

图 1 为所设计的硬件多线程处理器核的结构框图。在设计中对处理器的指令执行单元部分，例如译码电路、数字逻辑运算单元 ALU、指令寄存器 IR 等只分别设置一套。而对于每个线程在生存期全程都需要独立使用的工作空间，如通用寄存器组 GPR，PC 指针寄存器资源等，在处理器内部重复设置多组。这些重复设计的寄存器资源在处理器内部按页为单位划分为多组，并且在处理器中被命名为相同的名字，这些寄存器页面在处理器内部按硬件线程进行编号，处理器中的线程控制电路通过这些对应的编号实现调度管理。在此设计中，每个硬件线程在调入处理器执行的时候，处理器硬件将负责切换对应硬件编号的一页寄存器资源，每个硬件线程只可以访问到属于自己的那一页寄存器空间，加强了处理器对线程执行空间的保护。对于处理器来说，每个硬件线程使用的是不同的寄存器页面，但对于用户而言，看到的是相同名称的寄存器资源，这样在用户开发程序的时候，不必考虑实际使用的寄存器资源的具体分属页号，便于线程代码的开发。对于处理器本身，在处理硬件线程切换时，只需要直接将对应的寄存器页面编号改变就可以，既加速了处理器在处理器线程切换时的速度，同时也便于处理器设计的具体实现。

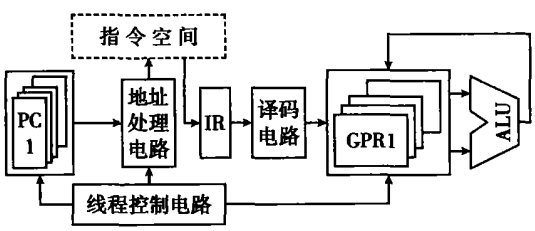

图 1　硬件多线程处理器结构框图

Fig. 1　Structure of hardware-based multi-thread scheduling processor

处理器在执行硬件切换的时候只需要按顺序完成“恢复工作空间”、“执行当前空间中的指令”、“保护当前的工作空间”三个步骤，即可以实现对每个线程的执行和工作空间的保护。

此外，与普通的单线程处理器的结构有所不同的是，由于处理器在多线程执行情况下，将同时涉及到多个线程的指令空间，因而在设计中，处理器按线程的执行顺序将多个线程的指令代码排入待执行指令队列中，处理器的指令寄存器 IR 从待执行指令队列中按顺序取出指令执行。在指令寄存器从待执行指令队列中取出某个线程的指令的同时，线程控制电路切换好此线程所使用的对应的通用寄存器组等资源，实现硬件线程的切换操作。此设计的另一个好处是，处理器的设计过程中，其指令译码、执行电路部分的设计与传统的单线程处理器的设计基本相同，有效地简化了处理器的逻辑设计。

2.3　线程选择模块

所设计的处理器中，每一个硬件线程都采用惟一的线程号来进行标识，在处理器内部，对硬件线程的区分所依靠的就是这些线程号。产生这些线程号的线程选择模块的结构原理如图 2 所示。

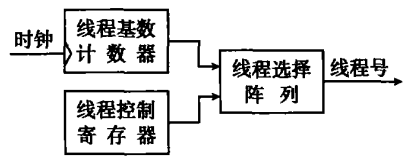

图 2　线程选择器结构原理图

Fig. 2　Schematic of thread-switching

线程选择电路具体由三部分构成：线程基数计数器、线程控制寄存器以及线程选择阵列。其中，线程控制寄存器记录当前被激活的硬件线程，通过这个寄存器可以提供给用户一个控制处理器中硬件线程的控制接口；线程基数寄存器中产生一个顺序的线程基数序号；线程选择阵列中包含一个二维的线性查找表。通过将线程基数计数器、线程控制寄存器两个部分的输出激励送入这个二维的线性选择阵列中，通过特定的二维线性查找关系即可以求出当前执行的线程号。

2.4　多线程下处理器工作空间的保护

处理器工作空间的保护主要包括在多线程运行环境下处理器对其内部各个寄存器、存储器的保护。在具体实现上有两种方法，一种是页面管理法，另外一种是资源快速切换法。

前一种方法的实现思路是：建立多个重复的需要保护的寄存器或存储器资源，处理器的硬件线程切换管理电路将这些寄存器、存储器按页为单位管理，而每个线程只能访问某个特定页中的资源。这种方法的特点在于，由于被保护的资源通过页面管理实现切换，不需要搬动其中的数据，因而在执行时速度较快，同时控制电路也比较简单。在设计中，对通用寄存器组、程序指针寄存器 PC 等寄存器资源的保护采用的是此种方法。后一种的设计思路在于：在处理器中额外设置一块高速静态存储器，当处理器需要切换资源时，利用硬件电路将要保护的寄存器中的内容存放到高速存储器当中去。设计中，对段寄存器以及处理器状态字

寄存器的保护采用的是此种方法。

在每条指令的执行周期中,并不是所有的寄存器都被全程使用的。例如,通用寄存器只有在 EX 和 WB 节拍才会使用。考虑到这点,可以通过适当排列操作顺序,保证处理器执行指令和处理器对寄存器资源的恢复、保护操作可以在同一个指令周期内完成。图 3 给出了多线程的运行环境下,程序指针寄存器 PC、处理器状态寄存器恢复、保护操作的时序图。图 4 给出了每个流水周期通用寄存器的读写操作时序示意图。其中图中的 CS 代表程序段寄存器,DS 代表数据段寄存器,SP 代表堆栈指针。

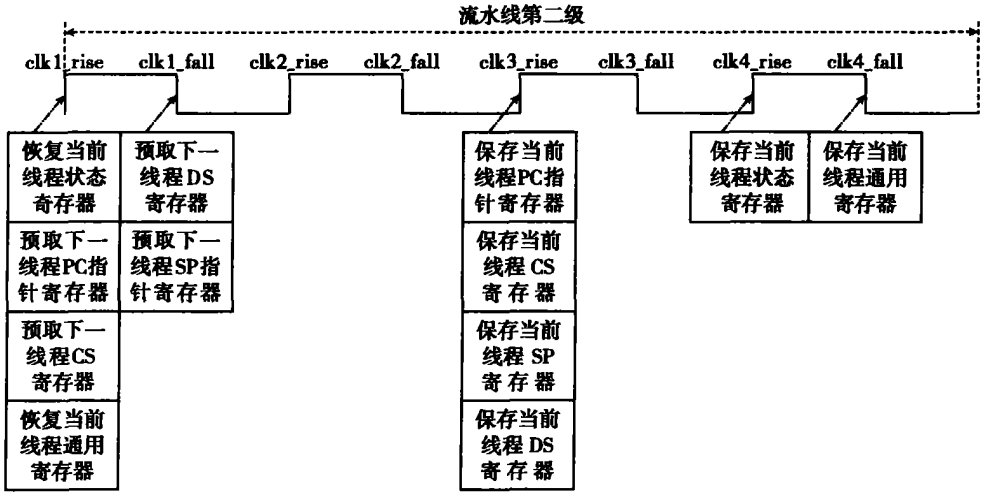

图 3 处理器工作空间的保护、恢复时序逻辑图

Fig. 3 Operating sequence of workspace protection

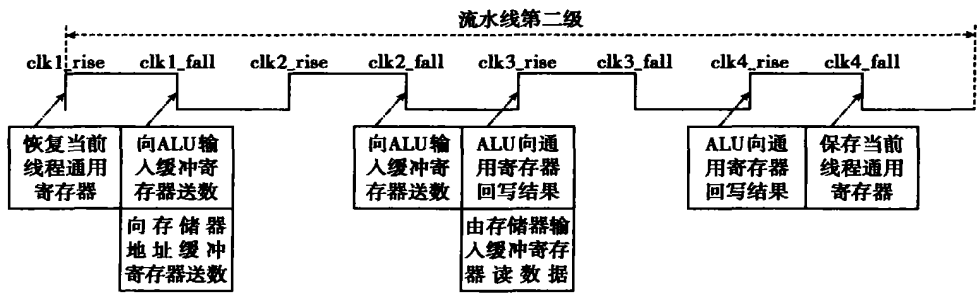

图 4 通用寄存器操作时序

Fig. 4 GRP operating schedule

3 实现及性能分析

本文所设计的硬件多线程处理器使用 VHDL 语言进行描述,并将其下载到 Xilinx 公司 XC2S200 型号 FPGA 芯片中进行实现。所实现的处理器的性能及参数设定如下:处理器的数据总线带宽为 8 位,处理器的工作主频为 20 MHz,处理器为 RISC 结构。指令集采用 ATMEL8515 处理器指令集的子集,最大支持 1 MB 的程序存储器空间及 1 MB 的数据存储器空间,最大支持 8 个硬件线程。处理器内部包含 32 个通用寄存器。

测试中所采用的两套参考系统分别为 WebitX 多任务嵌入式操作系统及 NRMX 多任务操作系统。其中,WebitX 操作系统是东大新业公司开发的一套在嵌入式系统上使用的抢占式多任务操作系统,此系统最大支持 8 个任务。硬件运行环境如下:ATMEL 公司 AT90S8515 RISC 单片机处理器,处理器工作频率 8 MHz,系统配置有 4 KB 程序存储器空间[8,9]。NRMX 操作系统是东大新业公司基于 Intel RMX51 操作系统改进的一套嵌入式多任务操作系统,此系统最大支持 8 个任务,采用时间片轮询方式执行。系统的硬件配置如下:处理器为 Intel 公司 i8051 处理器,工作频率 12 MHz,系统配置 12 KB 程序存储器空间。由于上述三套系统的处理器的机器指令码、指令执行所需时钟周期数以及处理速度各不相同,直接比较绝对的处理时间没有意义。因而在分析中采用比较用户线程执行比率的办法,执行比率定义如下:

$$\text{执行比率} = \frac{\text{多线程执行总时间}}{\text{单个线程执行总时间}}, \quad (1)$$

$$执行效率 = \frac{系统支持最大线程数}{执行比率}。\quad (2)$$

其中，多线程执行总时间指的是在被测系统中同时启动系统所支持的最大数量测试线程，并执行完这些线程所消耗的总时间。单个线程执行总时间指的是在被测系统中完整运行一次测试线程所消耗的时间。

三套系统在测试中所执行的测试线程为对应汇编指令编写的循环加法任务，测试结果见表 1。

表 1　测试结果

Table 1　Test results

测　试　项　目	硬件多线程处理器	WebitX	NRMX
单线程执行总时间/μs	61.5	63.1	308.1
8 线程执行总时间/μs	497.2	763.3	4171.6
执行比率/%	8.1	12.1	13.6
执行效率/%	0.99	0.66	0.59

从表 1 中可以看出，对于硬件多线程处理器系统，由于线程的切换是由处理器硬件来完成的，并不需要执行线程切换程序指令，因而在 8 个线程同时执行的情况下，执行效率较高，总运行时间基本等于一个线程运行时间的 8 倍。而对于操作系统软件实现的线程调度，由于需要执行线程切换调度处理程序，在实际运行中将额外消耗较多的处理器时间，因此执行效率将小于 1。

4　结　　论

本设计所实现的硬件多线程处理器在多线程条件下有较高的执行效率。但对于一个多线程处理系统的性能评价，还需要综合许多因素，例如处理器价格、处理速度的提升空间、功耗以及指令集兼容性等。因此，本设计的目的在于提出并尝试一种通过硬件控制完成多线程调度的方法，由于现今处理器设计实现技术以及 EDA 技术的局限性，对于真正实用化的硬件多线程处理器的实现，还有很长的路要走。但随着处理器设计技术的发展，以及用户对多线程处理能力需求的增加，在很多处理系统尤其是在嵌入式系统与单片机系统中，硬件多线程处理器将会有广阔的应用前景。

参考文献：

[1] Mano M M. *Computer architecture*[M]. 3rd ed. Englewood Cliffs: Prentice-Hall International Inc, 1998. 489－500.

[2] 韩光洁，王金东，林涛。基于 Web 管理的 Embedded Web Server 研究与实现[J]。东北大学学报(自然科学版)，2002, 23(11): 1021－1024。
(Han G J, Wang J D, Lin T. Design and realization of embedded web server based on web management[J]. *Journal of Northeastern University*(*Natural Science*), 2002, 23(11): 1021－1024.)

[3] Hennessy L. *Computer architecture: a quantitative approach*[M]. 3rd ed. San Francisco: Morgan Kaufmann Publisher, 2002. 60－84.

[4] Heuring V P. *Computer systems design and architecture*[M]. Menlo Park: Addison Wesley Longman, 1997. 70－81.

[5] Parhi K K. *VLSI digital signal processing systems design and implementation*[M]. New York: John Wiley and Sons Inc, 1998. 332－338.

[6] Armstrong J R, Gray F G. *VHDL design representation and synthesis*[M]. 2nd ed. New York: Prentice-Hall International Inc, 2000. 150－162.

[7] Mano M M. *Logic and computer design fundamentals*[M]. 2nd ed. New York: Prentice-Hall International, 2001. 161－194.

[8] Kuhnel C. *AVR RISC microcontroller handbook*[M]. St Louis: Butterworth-Heinemann Inc, 1988. 265－276.

[9] 耿德根。AVR 高速嵌入式单片机原理与应用[M]。北京：北京航空航天大学出版社，2001. 125－136。
(Geng D G. *The principle and application of AVR high-speed embedded MCU*[M]. Beijing: BUAA Press, 2001. 125－136.)

Implementation of a Hardware-Scheduled Multithread Processor for Embedded System

YIN Zhen-yu, ZHAO Hai, ZHANG Wen-bo, WANG Xiao-ying

(School of Information Science & Engineering, Northeastern University, Shenyang 110004, China. Correspondent: YIN Zhen-yu, E-mail: cmy @neuera.com)

Abstract: Taking account of the advantages and disadvantages of the existing software-based multithread scheduling for embedded system, a hardware-based design of multithread processor is proposed to implement simultaneously the multithread switching by polling through time slice. Thus, the scheduling cost for operating system can be minimized with the overall efficiency of the processor improved during the multithread execution, and the users' programming complexity can also be reduced under multithread conditions with the processor offering more powerfully protection for the threads during multithread running.

Key words: multithread processor; multithread processing; FPGA(field programmable gate array); embedded system; CPU design

(*Received December 7*, 2005)