适于 OMAP 的多级启动 Boot Loader

黄建尧,刘开华,李 琨
(天津大学 电子信息工程学院,天津 300072)

摘 要:通过对 OMAP 启动方式的分析,针对 OMAP 需要从外部 Flash 启动、耗时大、风险高的缺点,提出了一种多级启动的 Boot Loader 设计方案。该方案通过两级启动,在 RAM 中运行 Boot Loader,降低了代码运行的风险,减小了 Boot 过程的耗时。实验证明,使用该方法拷贝程序的耗时能够减少 20%左右。

关键词:Boot Loader;OMAP;多级启动
中图分类号:TP31 **文献标识码**:A

A multi-level Boot Loader for OMAP

HUANG Jian Yao, LIU Kai Hua, LI Kun
(School of Electronic and Information Engineering, Tianjin University, Tianjin 300072, China)

Abstract:To overcome the shortcomings of OMAP's external Flash booting mode, which will lead to much time consumption and high risk, a multi-level Boot Loader for OMAP is developed based on analysis of OMAP's booting mode. The multi-level Boot Loader runs in RAM through two levels booting which will cause lower risk, and the time consumption is about 20% less than before according to the experimental results.

Key words:Boot Loader;OMAP;multi-level booting

开放式多媒体应用平台(OMAP)是 TI 公司推出的,适用于下一代嵌入式终端设计的高集成度双核心处理器。它将一颗善于进行数据处理的 DSP 内核与一颗控制性能很强的 ARM 内核集成在一个芯片中,既改进了普通 DSP 处理器缺乏外部控制能力的缺点,也弥补了普通 ARM 处理器无法进行大运算量数据处理的不足。与单核心处理器相比,OMAP 双核结构处理器能够完成以前需要两颗处理器才能完成的工作,可以得到更低的系统功耗与成本,因而广泛应用于嵌入式终端中。

Boot Loader 是嵌入式系统的重要组成部分,它是在操作系统启动之前运行的初始化加载程序,用于设置操作系统运行的硬件环境并将 Flash 中存储的操作系统加载到 RAM 中运行。

1 OMAP 的启动方式

OMAP 多采用外部 Flash 启动方式。以 OMAP5910 为例,它包含一颗 TMS320C55x DSP 核心以及一颗 ARM925 核心,其中 MPU(ARM925)为主控核心,它可以控制 DSP 内核的启动、复位等操作。OMAP5910 上电时,MPU 首先启动,从地址 0x00000000 开始读取指令,运行程序。一般应在 0x00000000 地址处放置一条跳转指令,令其指向 Boot Loader 启动程序的入口处,OMAP5910 启动后跳转至 Boot Loader 程序处执行硬件环境设置及操作系统加载工作。根据 OMAP5910 的地址空间映射关系[1],0x00000000 地址属于 EMIFS(慢速片外存储器接口)总线,CS0 空间,用于连接外部 Flash,因而 OMAP5910 的启动阶段的程序在外部 Flash 中进行取指操作。

在实际应用中发现,如果操作系统占用空间比较大,使得 Boot Loader 需要从 Flash 中拷贝的代码量很大,那么系统的启动过程会耗费较长的时间。由于 MPU 需要在 Flash 中取指,又要将同一空间中其他部分的数据搬移到不同的 RAM(DSP 运行的内部 SRAM 以及 MPU 运行的 SDRAM)中,从而使得程序取指错误的概率增加。经过测试发现,随着 Flash 擦写次数的增多,Flash 性能下降,这种错误的概率会显著增加。这种错误体现到应用中,就会造成整个系统在启动过程中偶尔出现莫明其妙的死机,只能通过断电重新启动才能解决,因而这

种启动方式存在较大的隐患。

2 多级启动 Boot Loader

2.1 设计思想

考虑到启动过程中 MPU 需要从 Flash 中取指,这种操作既有风险且取指周期又长,因而应尽量减少在 Flash 中的取指动作。而 RAM 作为高速存取设备,其读写周期比 Flash 要短得多,因而将代码拷贝过程中的指令放到 RAM 中进行取指操作,既能降低取指周期,又能降低载入系统时的风险。通过上述分析,可以设计一种多级载入的 BOOT 过程,首先用一段尽量短的程序将传统意义上的 Boot Loader 拷贝到一段空白(载入操作系统时不会占用)的 RAM 中,之后再跳转到 RAM 中的 Boot Loader 程序入口,执行真正的 Boot Loader 程序。与操作系统相比,Boot Loader 程序代码要短得多,拷贝时间基本可以忽略,因而运行风险也小很多。这段程序在 Flash 中运行,任务是拷贝 Boot Loader 程序,可以称之为 Mini Loader。

2.2 具体过程

多级 Boot Loader 启动过程分为 Mini Loader、System Loader 以及 Flasher 3 个部分。

(1) Mini Loader

Mini Loader 在 Flash 中运行,其目的是从 Flash 中将 Boot Loader 程序拷贝到 RAM 中。为了尽量减小 Flash 中运行程序的数量就需要 Mini Loader 尽量短小,故而 Mini Loader 在进行拷贝动作之前只需要配置与外部总线 EMIFS、EMIFF 相关的寄存器。具体流程如图 1 所示。同时还要注意需要设置 MPU 的中断向量表 0x00000000 的跳转地址为 Mini Loader 的程序入口。

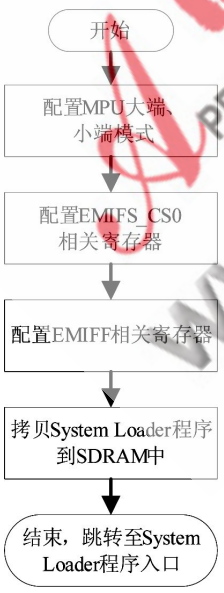

图 1 Mini Loader 流程

(2) System Loader

该部分为传统意义上的 Boot Loader 过程,完成硬件环境相关设置及初始化,并将操作系统载入到 RAM 中运行。需要注意的是,MPU 本身的中断向量表须载入到内部 SRAM 中,其他操作系统部分可以载入到外部 SDRAM 中。

由于 OMAP 平台为双核心结构,因此 System Loader 除了需要载入 MPU 系统本身之外,还需要配置 DSP 的载入及启动过程。DSP 有 3 种启动方法:Flash 引导方式与 ARM 载入方式,参考文献[2]对这两种方式进行了对比分析。本设计采用 ARM 载入方式,这样可以通过ARM 动态配置 DSP 部分的代码,控制 DSP 核心的处理流程。与参考文献 [2] 所不同的是,本方法中 DSP 程序采用与 MPU 操作系统相同的方式存储,也存入 Flash 中,而不是作为常量数组编入 Boot Loader 程序中,这种方式较参考文献[2]中的方法更加便于管理,单独对 DSP 程序进行升级也更加方便,而且减小了 System Loader 的程序长度,更有利于多级启动方式。

因为 OMAP 本身的 GPIO(MUPIO)有限,所以大多设计都要采用 FPGA 作为外围扩展控制器,用来扩展更多的控制端口以及通信端口。一般 FPGA 本身无法存储程序,其程序存储在片外 ROM 中,FPGA 上电之后可以通过多种方式自行加载到 ROM 中运行。本设计通过 MPU 加载 FPGA 程序,并将 FPGA 固件程序也存储于同一 Flash 中,这样省去了一片 FPGA 专用 ROM,既方便代码管理,又降低了硬件成本和设计复杂度。以 XILINX 公司的 XC2S 系列 FPGA 为例,其加载程序时可采用串行加载模式,选择 FPGA 为从模式,将 OMAP 本身的 MCBSP 配置为 SPI 主模式,工作于时钟停止模式,包含一个时钟周期延时,输出时钟信号高有效,采用帧同步模式,每个数据帧 8 bit。给 FPGA 载入程序时,要将存储于 Flash 中的 FPGA 程序代码读出,由于 XILINX 开发工具 ISE 输出的 HEX 文件以字节(8 bit)为单位,而 Flash 中存储的内容以双字(32 bit)为单位(由 ARM 指令长度而定),因而从 Flash 中读出 32 bit 的数据之后需要按照从高到低的字节顺序发送。而且,ISE 的 HEX 文件每字节 bit 排列顺序与 MCBSP 的传输顺序正好相反,因此每个字节内部需要将 bit 反向重排。

System Loader 流程如图 2 所示,启动顺序与本小节所述顺序正好相反,这是因为 FPGA 一般多用于进行硬件设备的控制,所以需要最先启动;而 DSP 由于其自身特点,多用于数据处理工作,作为 MPU 的协处理器使用,因此需要在 MPU 操作系统启动之前做好准备。

(3)Flasher

Flasher 过程主要负责向 Flash 中烧写编译好的程序目标码,主要包括 Boot Loader、MPU、DSP、FPGA 等部分。采用 RS232 串行端口与主机相连,实现程序目标码的下载。传输协议采用大多数编译器都支持的 Intel Hex 格式[5]。Intel Hex 格式采用 ASCII 字符表示方式,这样程序目标码的每个字节需要用 2 个 ASCII 字符来表示,包含

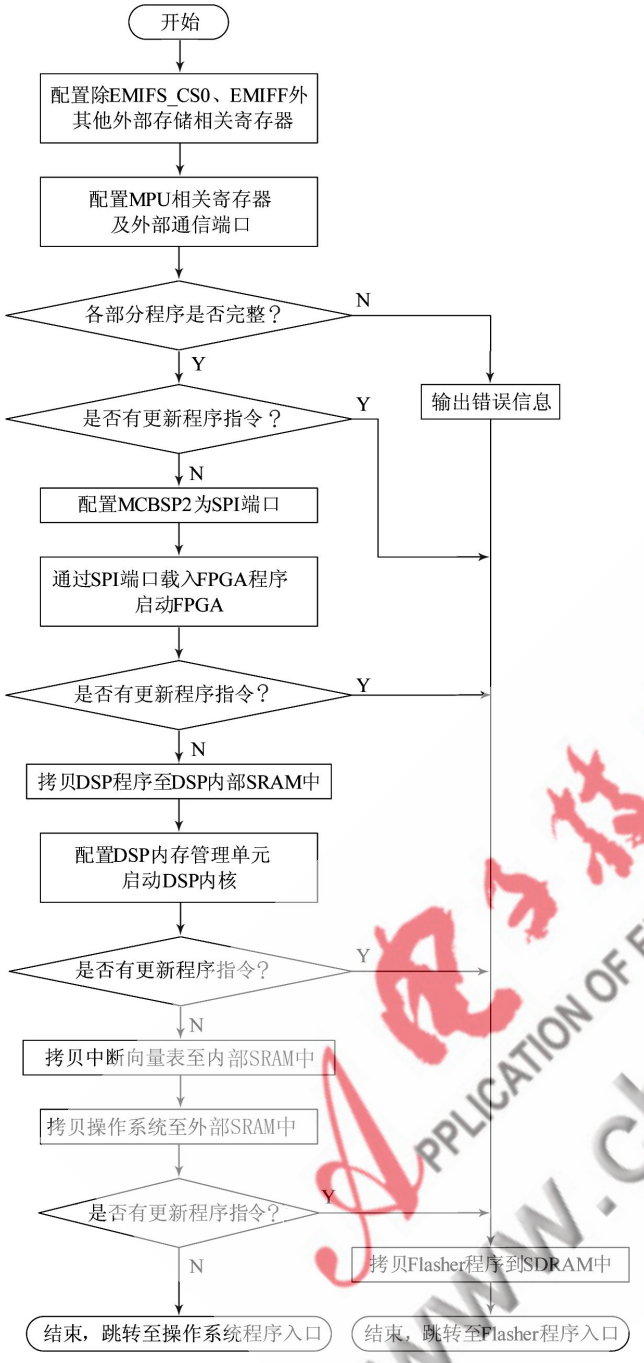

图 2 System Loader 流程

冗余信息较大，而且 Intel Hex 格式仅含目标码的地址信息与具体内容，而无法区分目标码的类别，因而需要对 Intel Hex 进行扩展，以降低冗余并支持更多的操作。扩展的 Hex 不使用 ASCII 字符，而直接使用原数据格式，这样可以降低一半数据量。为了与 Intel Hex 格式有所区分，采用“;”作为前缀标识，格式如表 1 所示。同时对数据类型进一步扩展，用以区分各种目标码类别，如表 2 所示。

可以看出，表 2 中仅包括擦除各种类型目标码的指令，而没有包含写入目标码时区分类别的具体指令，这是因为写入时各种不同类型的目标码可以直接通过地址来区分，每种类型目标码都有自己单独的地址段，相互之间没有交叉。Flasher 流程如图 3 所示，通过解析 Hex 记录格式，来判断命令类型，从而进行相应的操作。其中校验模式用于验证 Flash 中所存储的程序代码是否与串行端口上收到的数据一致。所有程序更新结束之后，通过重新启动命令复位 OMAP 芯片，使整个系统重新启动。

表 1 扩展 Hex 记录格式

标识码	长度	地址偏移	类型	数据	校验和
;	1 字节	1 字节	1 字节	*N* 字节	1 字节

表 2 扩展 Hex 记录类型

值	描述
00	数据
01	文件结束
04	基地址
10	擦除 Boot Loader 段 Flash
11	擦除 MPU 段 Flash
12	擦除 DSP 段 Flash
13	擦除 FPGA 段 Flash
14	进入校验模式
15	退出校验模式
16	重新启动 OMAP

3 拷贝程序耗时对比测试

采用 Intel 公司 RD48F3000P0ZBQ0 Flash 存储器以及三星 K4M56163PG 移动版 SDRAM 与 OMAP5910 相连，组成 OMAP5910 运行所需的最小系统，OMAP 运行频率 144 MHz。分别采用传统启动方式(程序在 Flash 中运行)以及本文所述的启动方式（程序在 RAM 中运行)从 Flash 中拷贝相同长度的数据到 SDRAM 中，测试其耗时。每次拷贝数据量选取 64 KB~8 192 KB，每种数据量每种方式测试 5 次，具体时间如表 3 所示，其中表 3(a)为程序在 RAM 中运行时的耗时，表 3(b)为程序在 Flash中运行时的耗时。由表中数据对比可以看出，拷贝数据量较大时，本文中所述的 RAM 拷贝方式优势比较明显，有大约 20%的提高，对于目前的嵌入式操作系统来说，基本内核的代码量都比较大，因而采用本文所述的方式能够带来比较大的耗时改善。

OMAP 作为一种整合了 ARM 控制能力与 DSP 数据处理能力的双核心处理器已经广泛应用于各种嵌入式设备中，但大多数 OMAP 处理器通过外部 Flash 启动，既浪费时间又有较高风险。本文针对 OMAP 启动过程中的这一缺陷，设计的多级启动 Boot Loader 较之传统方式有较大的性能改善，目前已在数字集群手持终端、网络多媒体可视电话等项目中得到成功应用，并取得很好的效果。

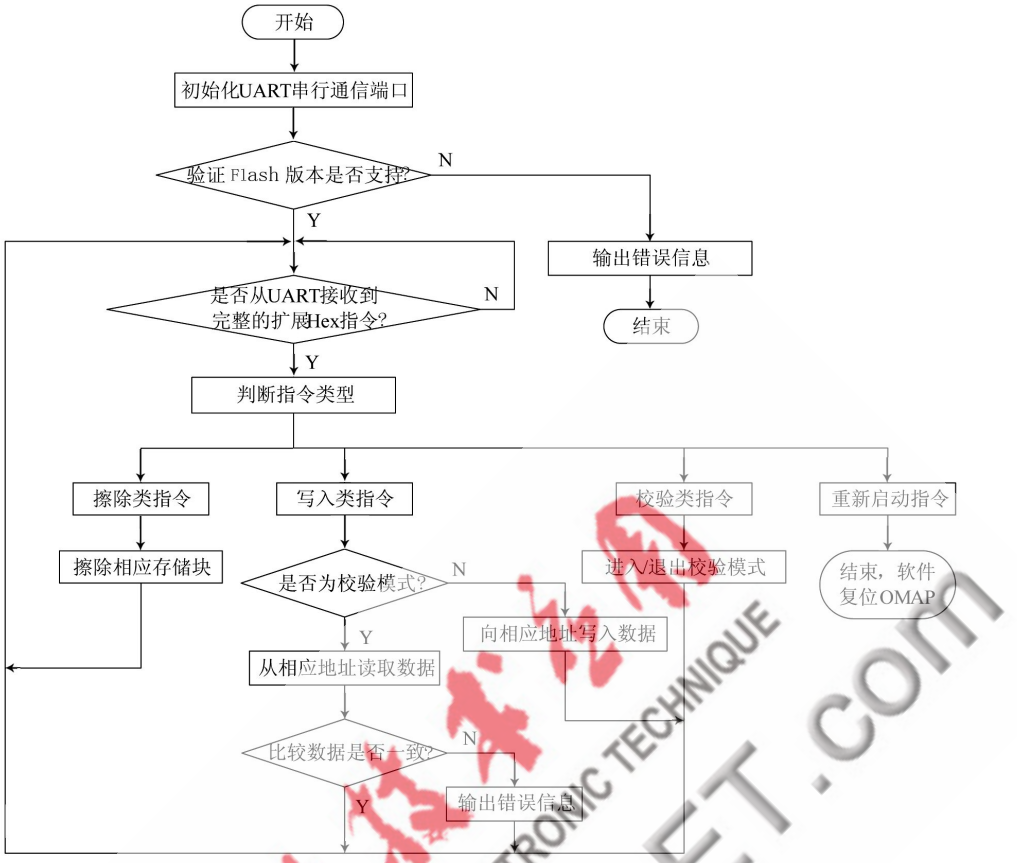

图 3 Flasher 流程

表 3 拷贝程序耗时测试

(a) RAM 拷贝方式

数据量/KB / 耗时/ms / 次数	64	128	256	521	1 024	2 048	4 096	8 192
1	15	31	63	125	234	469	938	1 890
2	16	30	63	125	234	469	937	1 891
3	15	30	63	125	234	469	940	1 889
4	15	32	64	125	235	469	939	1 891
5	16	31	63	124	236	469	937	1 891
平均	15.4	30.5	63.2	124.8	234.6	469	938.2	1 890.4

(b) Flash 拷贝方式

数据量/KB / 耗时/ms / 次数	64	128	256	521	1 024	2 048	4 096	8 192
1	16	31	78	142	297	594	1 172	2 344
2	15	31	79	140	297	594	1 172	2 343
3	16	31	78	141	297	593	1 172	2 344
4	16	31	78	139	298	594	1 172	2 343
5	16	31	78	141	297	593	1 172	2 344
平均	15.8	31	78.2	140.6	297.2	593.6	1 172	2 343.6

参考文献

[1] Texas Instruments OMAP5910 dual-core processor technical reference manual. 2003.

[2] 吕喜在,苏绍璟,黄飞,等,开放式多媒体应用平台 OMAP5910 双核程序装载方法[J]. 微计算机信息,2005,21(9-2):94-95.

[3] 李兴红.基于 OMAP 的嵌入式系统开发[J].数据采集与处理,2008,23:181-184.

[4] 李毅,李连云,张伟宏等.Bootloader 面向不同结构 Flash 的实现[J].计算机工程,2008,34(4):82-86.

[5] Intel Corporation. Intel hexadecimal object file format specification revision A. 1988.

[6] 张石,常皓,余黎煌,等. 基于 OMAP5910 的移动媒体播放机设计[J]. 电子技术应用,2007,33(2):27-29.

(收稿日期:2009-10-14)

作者简介:

黄建尧:男,1981 年生,博士研究生,主要研究方向:软件无线电。

刘开华:男,1956 年生,教授、博士生导师,主要研究方向:软件无线电等。

李琨:男,1981 年生,博士研究生,主要研究方向:通信系统。