第32卷 第21期 计 算 机 工 程 2006年11月
Vol.32 No.21 Computer Engineering November 2006
·开发研究与设计技术· 文章编号：1000—3428(2006)21—0270—03 文献标识码：A 中图分类号：TP311.52

TS101 DSP Flash 引导程序的设计和实现

王 琤，王 燕，龙 腾
（北京理工大学电子工程系雷达技术研究所，北京 100081）

摘 要： 在 TigerSHARC 系统上运行的 DSP 程序，系统上电复位后需要加载到 DSP 的存储空间上，这是从事 DSP 系统开发不可缺少的环节。该文描述了 TS101 的 Flash 引导程序，并针对具体的 Flash 芯片给出了引导程序的设计思想和实现过程。

关键词：Flash；引导程序；TigerSHARC101(TS101)

Design and Implementation of TS101 DSP Flash Boot Program

WANG Cheng, WANG Yan, LONG Teng
(Institute of Radar Technology, Dept. of Electrical Engineering, Beijing Institute of Technology, Beijing 100081)

【Abstract】 The DSP procedures should be loaded to the memory space of DSP when the system is powered on or reset. The procedure is necessary to DSP system. The paper describes the flash boot program. According to specific flash chip, it gives the design concepts and implement of boot program.

【Key words】 Flash; Boot program; TigerSHARC101(TS101)

TigerSHARC(TS)系列 DSP 是 ADI 公司推出的高性能的数字信号处理器，该处理器广泛应用于第 3 代移动通信、雷达、声纳、工业仪器等领域。

对于 TS101 而言，支持 3 种方式加载：

(1)EPROM(Flash)引导：该引导方式为主引导，当系统在上电复位期间 $\overline{BMS}$ 的管脚为持续低电平则 DSP 会从处理板上的 EPROM(Flash)空间自动加载程序到 TS 的内部存储空间，从而完成程序的自动执行；

(2)主机引导：该种模式称为从模式引导，当系统上电复位期间 $\overline{BMS}$ 管脚为持续的高电平，使处理器进入空闲的状态等待外部主机或者其他的处理器进行程序的引导；

(3)链路口引导：该种引导模式称为从模式引导，当系统上电复位期间 $\overline{BMS}$ 管脚为持续的高电平，使处理器进入空间的等待状态，直到有某一个链路口对它进行程序的加载。

用 EPROM(Flash)进行程序加载是目前大多数嵌入式系统所选择的，主要是因为它的方便易行，现在的 Flash 可以通过 JTAG 对 Flash 空间进行直接的读写和擦除操作，而且一旦写入以后，只要不对它进行特定的操作，Flash 空间的内容就会保持不变，本文描述了基于 EPROM(flash)的引导模式。

1 TS101DSP 的 Flash 引导模式

任何处理器在开机时都需要复位，从而使得处理器核及片内其它功能部件都处于一个确定的初始状态，并从这个初始状态开始工作。复位后，处理器可以自动从外部设备加载程序，称为引导。ADSP－TS101S 的引导方式由复位时 $\overline{BMS}$ 管脚的电平高低决定。

在复位的这段期间 $\overline{BMS}$ 为低电平，则设置为 EPROM(Flash)引导方式，对于这种方式，在 $\overline{RESET}$ 信号无效后，$\overline{BMS}$ 信号就成为输出信号，作为 EPROM 的片选信号；如果其为高，ADSP－TS101 将处于空闲(IDLE)状态，等待外部主机或链路口来引导。

对于基于 TS101 开发的多处理器系统，它的加载可以分为 2 种情况：

(1)多片 DSP 按照它们的优先权轮流从 Flash 中加载数据，这种方式连接是多个 DSP 的 $\overline{BMS}$ 管脚都跟唯一的 Flash 的 CS 管脚相连，加载核对多个 DSP 的可执行文件*.DXE 生成加载文件，Processor0 为主 DSP 的可执行文件，其他依次按照系统的硬件设计添加就可以了。

(2)主 DSP 从 Flash 中加载程序，其他的 DSP 通过主 DSP 去加载。在这种情况下，只有主 DSP 的 $\overline{BMS}$ 管脚和 Flash 的 CS 相连，其他 DSP 的 $\overline{BMS}$ 管脚都通过一个上拉电阻连到 V_{DD_IO}，其他的 DSP 的引导可以通过 DSP 之间相连的 Link 口进行加载，此种情况下 ldr 文件的生成是独立的，主 DSP 的 ldr 文件的生成跟单处理器的一样，但是其他的 DSP 的 ldr 文件的则根据是通过 Link 口还是 Host 加载而选择不同的方式去成生加载文件，最终其他的加载文件以数组的形式存放到主 DSP 的 ldr 文件中。

当选用 EPROM(Flash)模式以后，TigerSHARC 初始化它的外部端口 DMA 通道 0 用来从 Flash 存储器向 TigerSHARC 的内部存储块 0(位于 0x00-0xFF 的 Block0)传输 256 个 32bits 的字代码，DMA 通道 0 的相应中断被初始化为 0。因此当 DMA 传输完成以后，TigerSHARC 继续从 0x00 开始执行程序。传入 Block0 的这 256 个字相当于是一个引导核，它用来初始化 TigerSHARC 其他存储空间。ADI 公司提供了默认的加载核的源文件 TS101_prom.asm。

该引导核和原代码的可执行文件*.exe，在引导加载程序 elfloader.exe 的作用下生成加载文件*.ldr，这个 ldr 文件的头 256 个 32bits 字就是前面提到的加载核，生成加载文件的过

作者简介：王 琤(1974－)，男，博士生，主研方向：高速实时信号处理的软硬件设计和实现，雷达信号处理，大规模软件体系结构的设计和开发等；王 燕，硕士生；龙 腾，教授、博导

收稿日期：2005-11-18 **E-mail：**wang_jacky@163.com

程可以用图 1 表示。

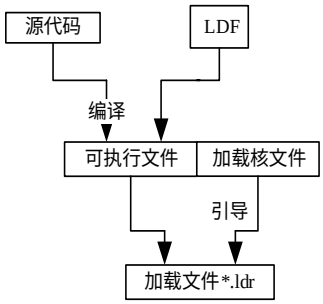

图 1 生成加载文件的过程

2 Flash 引导程序的设计和实现

往信号处理板的 Flash 空间加载程序，首先要做的是验证这个 Flash 芯片的的好坏，然后根据该系统是单处理器还是多处理器，生成不同 ldr 文件来加载程序。

验证一个 Flash 芯片的好坏，对它所进行的操作是，往该 Flash 空间中写入数据，然后在读出数据，如果能写能读则说明该 Flash 芯片是好的，可以进行程序的加载。Flash 寄存器不是 TS 的内部寄存器，它们之间没有地址映射关系，只有通过连接，把 Flash 芯片连到 TS 外部的某一存储地址空间，然后通过对该地址空间的读写操作而达到对 Flash 的读写操作。在笔者开发的 TS101 信号处理系统中采用的 Flash 芯片是 AT29LV040A，以该 Flash 为例对其主要的几个特性进行说明：

(1)2 048 个扇区每个扇区 256B；

(2)8 位数据总线 19 位的地址总线；

(3)2 个 16KB 的加载块。可以锁存 Flash 空间的前 16KB 和后 16KB，一旦该区域被锁，则不能通过任何方法解锁，确认该区域是否被锁的方法是在 Flash 存储器件处于软件产品的识别模式的时候如果 Flash 地址空间的 00002H 和 7FFF2H 空间的内容是 FE 则说明该块没有被锁，如果内容是 FF 则说明该块被锁，00002H 代表的开始部分的 16KB 的加载块，7FFF2H 表示的是结尾处的加载块，软件产品识别模式通过如下 3 个操作得到：

1)把数据 AAH 写入 Flash 的地址空间 0x05555H；

2)把数据 55H 写入 Flash 的地址空间 0x2AAAH；

3)把数据 90H 写入 Flash 的地址空间 0x05555H。

延迟大概 20ms 的时间以后，进行读数，地址 0x00000H 的内容就是制造商的代码，地址 0x00001H 的内容就是产品的 ID 号。从软件产品识别模式转换到标准操作模式需要经过以下几个操作：

1)把数据 AAH 写入 Flash 的地址空间 0x05555H；

2)把数据 55H 写入 Flash 的地址空间 0x02AAAH；

3)把数据 F0H 写入 Flash 的地址空间 0x05555H。

(4)软件数据的保护特性，该特性主要是说一旦程序或者数据被写入到 Flash 存储器的存储空间如果不进行特殊的软件操作该数据和程序是不会被修改的，即使进行了软件的读写操作，所作做的特殊操作包括如下 3 个部分：

1)把数据 AAH 写入 Flash 的地址空间 0x05555H；

2)把数据 55H 写入 Flash 的地址空间 0x02AAAH；

3)把数据 A0H 写入 Flash 的地址空间 0x05555H。

就一般情况而言，往 Flash 空间写数的时候都需要经过这 3 步的操作，但是对于从来没有使用过的片子第一次写数的时候不需要经过这 3 步操作。

该 Flash 为 19 位地址总线，接到 TS 的低 19 位地址空间，设计硬件系统的时候把它映射到了 TS 的 MS0 空间，所以 TS 访问 Flash 器件的时候的地址操作空间是 0x8000000。同时由于 Flash 是一个慢速器件，因此对它进行访问的时候要根据时序对 TS 总线配置寄存器 SYSCON 进行配置。由于 Flash 芯片的数据线是 8 位，它直接连到 TigerSHARC 的 Data0－Data7，因此要把程序加载到 Flash 的存储空间所要做的必不可少的一项工作就是要把 TigerSHARC 的 32 位数据拆成 8 位存在一个数组中。根据 Flash 型号的不同，它的数据线的位数也不一样，存储容量也不同，不同形式的 DSP 数据线的连接也不一样，这个要根据具体的情况做具体的变换。以本系统所用的 Flash 为例拆分数据的形式如图 2 所示。

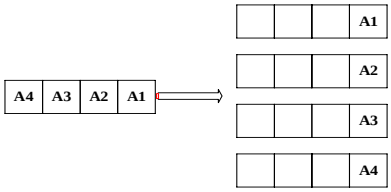

图 2 Flash 拆分数据的形式

不管是什么型号的 Flash 对它的读写操作都是每次一个扇区即 256B。验证部分的流程如图 3 所示。

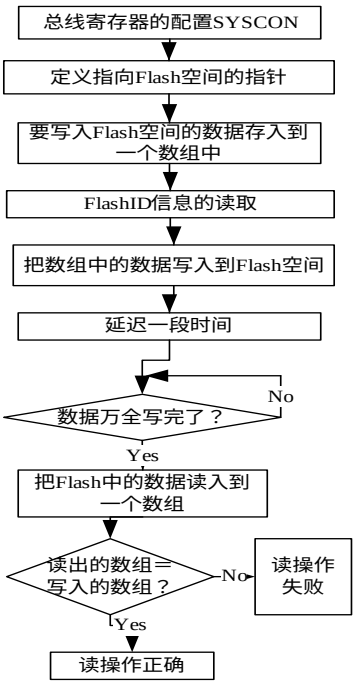

图 3 验证 Flash 功能

验证一个 Flash 芯片的具体操作可以分为如下几个步骤：

(1)配置总线寄存器 SYSCON，Flash 映射的外部地址空间要插入 3 个等待状态；

(2)定义一个指向 Flash 地址空间的指针；

(3)把要写入 Flash 空间中的数据存入到一个数组中；

(4)FlashID 号的获取，以确认该 Flash 的两个锁存块是否能够读写数据，只有当第一个引导块没有被锁时，Flash 中的程序才能正确地引导，即使第一块被锁也一样能够验证 Flash 的读写，但使 Flash 空间中的程序不能正确的引导；

(5)把数组中的内容写入到 Flash 的内存空间，可以利用对一个地址空间的普遍的读写操作的方法，也可以利用 DMA 的传输方式，由于 Flash 响应数据会有一个时间间隔，因此

在数据写入到 Flash 空间以后要延迟一段时间，为了增加每次操作的可靠性，在下一个操作之前应该判断上一次操作是否结束；

(6)把刚刚写入 Flash 的内容读出到一个数组，验证这个数组和刚刚写入数据的数组的内容是否一致，如果一致则说明读写操作是正确的。

如果经过上面的验证步骤证明 Flash 能够进行读写操作，接下来所要做的操作就是生成 ldr 文件和把生成的加载文件写入到 Flash 空间中去。关于单处理器和多处理器不同的地方就是加载文件的生成方式不同而已。

在 Visual DSP++的开发环境中，ldr 文件的生成过程可以分为以下几个步骤：

(1)在 Visual DSP++开发环境的界面中，选择 project 菜单下的 project option 选项，Type 的类型在下拉中选择 DSP loader file，Name 为所需要加载的工程的名字，setting for 为所要生成的 ldr 文件的存放路径；

(2)点击 load 标签，其中 Kernel Files 为加载核文件，默认的就是 Flash 的加载核文件，如换作另外两种主机或者链路口的加载则需要重新定位加载核文件，在安装目录的 ldr 文件夹下找到所要的*.exe 文件，Output File 为所要生成的 ldr 文件的名称，不需要后缀，若不写则默认的文件名称和工程同名，Boot Type 为加载类型的选择，Prom、Host 和 Link 选择相应的加载类型 Kernel File 处的加载核就会改变成相应的核文件，Format 为加载文件生成类型的选择，Multiprocessor 为多处理器的选择，对于单处理器系统该项不用选择，但是对于多处理器系统该项需要选择，根据所使用处理器在对应处理器标号处添加相应的可执行文件的名字*.exe；

(3)按照自己的需要进行设定以后点击确定按钮，然后编译程序，则此时在 setting for 的目录下生成所需要的 ldr 文件。把 ldr 文件写入到 Flash 空间的步骤如下：

1)配置总线寄存器 SYSCON；

2)把 ldr 文件读入到一个数组中，由于 Visual DSP++是 32 位的数据总线，读入的数据会占满 32 位的每一位，所以要把这个数组经过变换把数据存到另外一个数组中去，该数组的大小是原数组大小的 4 倍；

3)由于 ldr 文件太大，如果把数据存入到一个全局数组中则会造成该数组的大小超过 Visual DSP++的内存空间，又由于 Flash 存储器每次写入必须是一个扇区，因此根据这些特性，把存放文件的数组按一个扇区的大小为 256B 来划分，而实际上每一个 256B 写入到 Flash 中就是 4 个扇区的大小，对每一个扇区的操作按照验证 Flash 特性时写 Flash 来操作就可以了；

4)把整个数据都写入到 Flash 中以后，为了保证写入数据的正确性，最好把 Flash 中的数据导出到一个数组中，用以和写入的数据的数组进行验证，如果没有错误则说明写入的数据是正确的。

ldr 文件大小的计算定义为由 Visual DSP++生成的加载文件*.ldr 写入到一个数组中的大小的计算。用编辑器看 ldr 文件，生成的格式用二进制，每行 16 个数，每页 36 行，计算该文件一共有多少行，每行的 16 个原素按 4 个计算，因为在存入到一个数组的过程中，Visual DSP++的开发环境中是 32 位的数据总线，每个数组能放下 4 个数，所以需要的数组的大小为总行数×4。由于 Flash 空间是 8 位数据总线，因此还要把该数组拆成一个新的数组，该数组的每个元素虽然也占据 Visual DSP++的 32 位数据总线但是每个数组只有 8 位，所以最后要写入到 Flash 存储空间的数组大小为刚刚计算的数组大小的 4 倍。

把 ldr 文件写入到 Flash 存储器中的实现如图 4 所示。

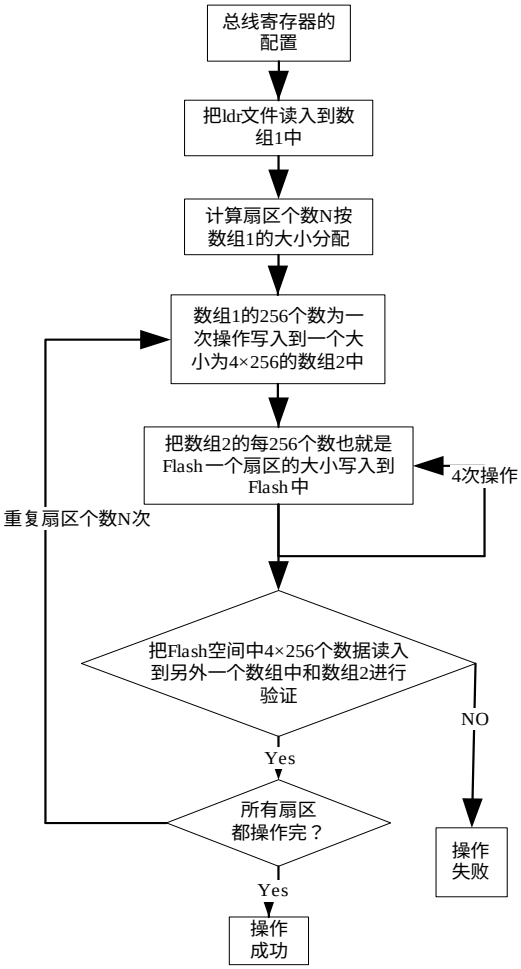

图 4 ldr 文件写入 Flash 流程

3 结论

本文描述了 TS101 的 Flash 引导模式和开发引导程序的过程，针对具体的 Flash 芯片给出了详细的引导程序的设计和实现过程，该方法在基于 3 片 TigerSHARC101 的 DSP 系统中得到了应用。

参考文献

1 刘书明. TigerSHARC DSP 应用系统设计[M]. 北京: 电子工业出版社, 2004.

2 Analog Devices Inc.. TigerSHARC DSP Hardware Reference Re-vision 1.0[Z]. 2003

3 Analog Devices Inc.. ADSP-TS101 TigerSHARC Processor Prog-ramming Reference[Z]. 2003.

4 Analog Devices Inc.. VISUAL DSP++ 3.5 Getting Started Guidefor 32-Bit Processors(Revision 1.0)[Z]. 2004.

5 Lerner B. ADSP-TS101S TigerSHARC Processor Boot Loader Kernels Operation[EB/OL]. http://www.analog.com, 2003.