基于 NAND Flash 的转译层的设计

侯 超,徐建城

(西北工业大学,陕西 西安 710129)

摘 要:基于 NAND Flash 的嵌入式存储系统以其轻巧便携、读写速度快等特点成为当前嵌入式存储系统的主流配置。但由于固有坏块以及在擦、写过程中随机产生的坏块影响了 NAND Flash 的实际应用,所设计的 NAND Flash 的驱动转译层具有坏块管理机制并实现上层文件系统的连续读写功能。

关键词:NFTL;地址映射;坏块管理

中图分类号:TP333.96 **文献标识码**:A **文章编号**:1674-7720(2010)24-0073-04

Translation layer design based on NAND Flash memory

HOU Chao, XU Jian Cheng

(Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: NAND Flash-based embedded storage system for its characteristics of lightweight portability and fast read and write become the mainstream configuration of the embedded storage system. Due to the inherent bad blocks and generation random bad blocks during erase or program operation, which affects the practical application of NAND Flash. In this paper, a flash translation layer of device driver is designed, which has bad block management and implements a continuous read and write capability for file system.

Key words: NFTL; address remapping; bad block management

目前市场上闪存芯片主要有两类,即 NAND Flash (Not And Flash ROM)和 NOR Flash(Not Or Flash ROM)。前者具有容量大、读写速度快、芯片面积小、单元密度高、擦除速度快、成本低等特点,更适合于大批量数据存储的嵌入式系统。如今 Windows 仍是桌面系统的主流,对 FAT 文件系统提供了天然的支持。然而就技术而言,FAT 文件系统并不适合 Flash,因为 Flash 设备并不是块设备[1],为了不破坏兼容性,并在 NAND 型闪存中应用 FAT 文件系统,国际上提出了闪存转译层 FTL(Flash Translation Layer)的解决方案。

1 NAND Flash 嵌入式存储系统结构

基于 NAND Flash 的存储系统的设计首先要解决坏块问题。由于 NAND Flash 自身存在固有坏块并在擦除和编程中又随机产生坏块,因此为了提高设备的可靠性应该将这两种操作分散在闪存不同的块中,以避免对某块的过度操作。

一般的基于 NAND Flash 嵌入式存储系统驱动结构分为三个层次:最底层是硬件操作接口,负责将主控芯片与 Flash 的控制管脚相连,这方面的固件主要实现对 NAND Flash 的物理操作;中间层是闪存转译层 NFTL (NAND FTL),是封装在 Flash 驱动中的软件模块,其作用是将 Flash 模拟成与磁盘相类似的块设备,使对上层操作系统而言,NAND Flash 就像普通磁盘一样被访问。这一层主要是封装一些特殊的复杂管理控制功能;最上面的层就是文件管理层,功能类似于普通磁盘上的通用文件系统,向上层提供标准的文件操作接口。基于 NAND Flash 的嵌入式系统存储结构原理图如图 1 所示。

根据以上两个方面,既要在驱动中实现坏块管理,又要进行块模拟,所以可用的方法有两种[2]:一是在上

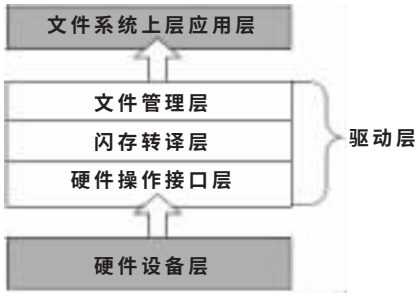


图 1 基于 NAND Flash 嵌入式系统存储结构

层文件系统中解决坏块问题，驱动层只实现本身的功能，文件系统为驱动层提供不变的接口，为上层应用程序提供可靠透明的服务。这种方法较简单，开发周期比较短，但只对特定应用的嵌入式系统有很强的适应性；第二种方法是在驱动层的 NFTL 中解决坏块问题，将不可靠的 NAND Flash 虚拟成可靠的存储设备，为上层文件系统提供可靠透明服务，这种方法较第一种更复杂，但是此法具有较强的可移植性并能彻底断绝与文件系统的联系，其他文件系统也同样适用。

本文是以 Samsung 的 NAND Flash K9F2808U0C 作为存储芯片，设计了一种在 NFTL 上实现坏块管理并且实现连续数据读取的方法。

2 设计思想

2.1 闪存空间划分

K9F2808U0C 是 16 MB×8 bit 的 NAND Flash，共有 1 024 个 Block，1 Block=16 KB，32 Page/Block，1 Page=528 B=(512 B+16 B)，其中 16 B 为备用区，主要存放 NAND Flash 出厂坏块标记、ECC 校验码以及用户自定义区。K9F2808U0C 地址空间是 24 bit，分三个周期依次送入 NAND Flash 的地址锁存器。本文使用的地址均为字节地址，数据类型为 DWORD(4 B)。

将 K9F2808U0C 的存储空间划分为四个区：坏块映射表存放区、交换块区、坏块映射区和实际数据存放区。文件系统管理的空间就是实际的数据存放空间，如图 2 所示。

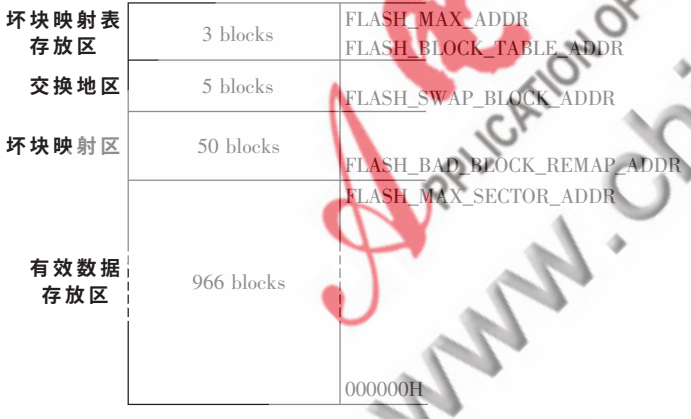


图 2 K9F2808U0C 存储空间划分

2.2 各分区宏定义

```
#define FLASH_BLOCK_SIZE    0x40000      //16 KB/Block
#define FLASH_PAGE_SIZE     0x200        //512 B/Page
#define FLASH_SECTOR_SIZE   0x200
                                  //1Page=1Sector(only K9F2808U0C)
#define FLASH_BLOCKS_TABLE  3//坏块映射表存放块数
#define FLASH_SWAP_BLOCKS   5        //交换区的块数
#define FLASH_BAD_BLOCKS_REMAP   50
                                  //坏簇重映区的块数
#define FLASH_MAX_ADDR   0xFFFFFF
                                  //Flash 最大字节地址
```

各分区首地址计算公式：

FLASH_BLOCK_TABLE_ADDR=FLASH_MAX_ADDR+1-3*FLASH_BLOCK_SIZE);

FLASH_SWAP_BLOCK_ADDR=(FLASH_BLOCK_TABLE_ADDR-5*FLASH_BLOCK_SIZE);

FLASH_BAD_BLOCK_REMAP_ADDR=(FLASH_SWAP_BLOCK_ADDR-50*FLASH_BLOCK_SIZE);

FLASH_MAX_SECTOR_ADDR=(FLASH_MAX_ADDR-3*FLASH_BLOCK_TABLE_ADDR-5*FLASH_SWAP_BLOCK_ADDR-50*FLASH_BAD_BLOCK_REMAP_ADDR);

文件系统管理的最大字节地址。

任意地址 Addr：

所在块地址：Addr&(~(FLASH_BLOCK_SIZE-1));

块内偏移地址：Addr&(FLASH_BLOCK_SIZE-1);

块中的页：(Addr&(FLASH_BLOCK_SIZE-1))/FLASH_PAGE_SIZE;

2.3 分区功能设计

坏块映射区存放复制 3 份的坏块信息 BBI(Bad Block Information)表。复制 3 份是预防系统突然断电，造成 BBI 表数据丢失。选择最后 3 个块，主要是出于固件设计。当 Flash 首次上电，固件程序通过读取 Flash ID，获得设备的容量等信息，然后从 Flash 的最后一块中寻找 BBI 表，如果最后一块没有发现 BBI 表，则认为此块为坏块，继续前移寻找，依此类推，直到在预留的 3 个块中找到，并将其数据读入到在主控芯片为其开设的 RAM 中。如果还找不到，则固件认为该片 Flash 没有 BBI 表。

交换块区是对 NAND Flash 进行擦除或写操作时用来临时存放数据，共分配 5 个块。选取 5 块是出于可靠性设计。用一个数组 FlashSwapBlockStatus[FLASH_SWAP_BLOCKS]记录交换块状态：有效还是已经损坏。初始化时，固件认为所有的交换块都是有效块，在随后对其进行擦除或写操作时，通过读 Flash 状态寄存器判断该交换块的真实状态，并记录在数组中。交换块的管理围绕固件请求返回当前可用交换块地址或当前正在使用的交换块地址，并判断标记当前使用的交换块状态为坏。

坏块映射区是当主机向数据区写数据时，检测到当前块(数据区)为坏块时，将数据写到坏块映射区中的相应好块中，并且将这两个块的块地址记录到 BBI 表中，以后主机若要对当前块(数据区)访问时，只需读 BBI 表就可以找到相应映射块，从而代替坏块的访问。这样就使文件系统所见逻辑块地址 LBA(Logical Block Address)变成连续的，但实际上物理块地址 PBA(Physical Block Address)可能并不连续。上述方法就是坏块管理的精髓。出于保守设计本文共选 50 块作为重映块。用数组 FlashRemapBlockStatus[FLASH_BAD_BLOCKS_REMAP]标识坏块映射区的状态：未使用、已使用还是已经损坏。初始化时认为坏块映射区中所有块都是好块。

3 NFTL 坏块管理设计

3.1 构建 BBI 表

用一数组 FlashBadBlockTable[2][FLASH_BAD_BLOCKS_REMAP]存放 BBI 表。第一维为坏块(数据区)的块地址,第二维为映射块地址(重映块)。

BBI 表的构建可在 Flash 首次上电时,通过读取厂商设置在 Flash 备用区中的坏块标记识别坏块,建立坏块映射表,此法初始化时间与 Flash 的容量成正比;或通过读 NAND Flash 所有块内容并和 0xFF 作比较[3],如果不相同,则表示坏块。这种方法只针对新闪存,并且构建 BBI 表的时间长,主控芯片的占用率高;另一种方法是在 Flash 初次上电时先不建立坏块表,认为当前所有的块都是好块,在随后操作中发现坏块,并更新 BBI 表。本文选择后一种方法。

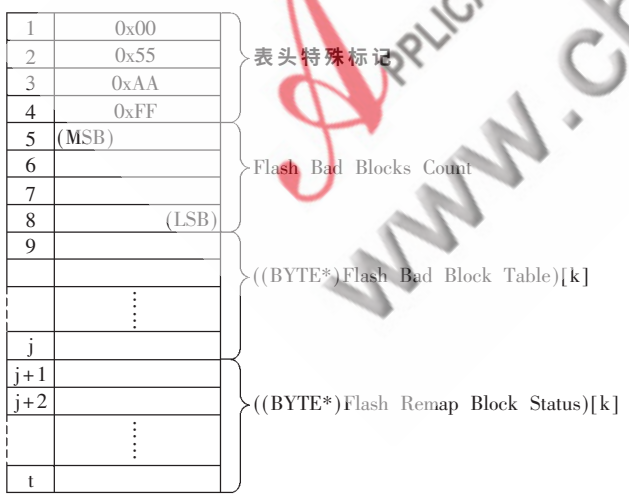
3.2 坏块映射表区的设计

将 BBI 表通过特殊的方式保存,以后通过此种方式的逆向来识别 BBI 表。坏块映射表区的 3 个块的设计方法如下:

(1)在每块最后一页的首字节处做特殊标记,用于标识该块有没有准备被擦除:0xFF 表示没有准备被擦除;0x00 表示该块已经准备被擦除[4]。

(2)在每块的倒数第二页的首字节处写 0x00,表示该块存放 BBI 表数据;并在第 2、3、4、5 个字节处以大端模式存放校验和,用于校验该块中存放的所有数据的正确性。该页的剩余字节写 0xFF。

(3)在第一页起依次写:表头特殊标记(0x0055AAFF)、坏块总数(FlashBadBlockCount)、BBI 表(FlashBadBlockTable)、重映块状态(FlashRemapBlockStatus),其结构如图 3 所示。



j=sizeof(Flash Bad Block Table[0][0])*FLASH_BAD_BLOCKS_REMAP*2+7
t=sizeof(Flash Remap Block Strtus[0])*FLASH_BAD_BLOCKS_REMAP+j-1

图 3 坏块映射表区结构

3.3 逻辑地址映射

在与主机批量数据传输时,首先对主机发送的 CBW 命令解析,从而固件获得主机请求数据传输的 LBA,再对 LBA 地址映射以此获得相应的 PBA,然后再执行主机命令。具体方法:首先读取当前总的坏块数 FlashBadBlocksCount,如果为 0,表示无坏块,LBA 就不用映射,直接返回;如果最后一次访问地址与 LBA 同属于一个块,那么也不用地址映射;如果当前坏块的数量为 1,判断 LBA 块地址与 BBI 表中坏块地址是否相同,从而决定是否采取地址映射;如果当前坏块的数量不为 1,就需要查找 BBI 表判断是不是含有 LBA 的块地址,如果存在,则要地址重新映射,如果不存在,就不用重新映射。可以采用二分法查表实现两地址快速比较[4]。

3.4 坏块处理

对映射后的地址进行写或擦除过程中发现当前块变坏的处理方法:首先查找映射区中的块状态 FlashRemapBlockStatus[FLASH_BAD_BLOCKS_REMAP],寻找可用的映射块。如果 50 个重映块都被标记为坏或已使用,则程序进入死循环;如果找到可用的映射块,则固件对可能出现坏块的三种情况进行散转:(1)擦除当前块时出错:将当前块地址与可用的重映块地址写入 BBI 表中,按地址大小排列有利于二分法查表,并返回重映块地址。(2)复制某页从交换区到操作地址时失败:首先获得当前使用的交换块的块地址,然后判断操作地址是在数据存储区还是在重映区。如果在重映区,说明当前操作地址所对应的在数据存储区中的块已经是坏块,并且在这次操作中重映块也变坏,此时就应该标记当前映射块状态为坏,并在重映区中寻找下一个可用重映块,将原来在数据存储区中的块地址与更新后的映射块地址写入 BBI 表中,并返回新的映射后的地址;如果在数据存储区,就进行第一次重映射,更新坏块表,并且将映射地址返回。然后再完成复制工作。(3)向操作地址写一页数据时出错:将当前地址所在块中的前面页从交换块中相应地址复制到重映块,然后将操作地址所在当前页中的没有写到的数据从交换块中复制,并将缓冲区中的数据重新写到重映射的地址中,并返回重映射地址。函数实现如下:

```
DWORD FlashDealBadBlock(DWORD Addr,DWORD Type)
{
 DWORD i;
 DWORD RemapBlockAddr;
 DWORD SwapBlockAddr;
while(1)
{
   RemapBlockAddr=FlashGetNewRemapBlock();
   if(RemapBlockAddr==-1)
      return Addr;
   switch(Type)
      { case 1:
        goto Exit;
        break;
```

```
            case 2:
            SwapBlockAddr=FlashGetCurrentSwapBlock();
        for(i=0;i<(Addr&(FLASH_BLOCK_SIZE-1))/
                                    FLASH_PAGE_SIZE+1;i++)
        {
        if(0x00==(FlashCopyPage(SwapBlockAddr+
                    i*FLASH_PAGE_SIZE,RemapBlockAddr+
                                    i*FLASH_PAGE_SIZE)))
            goto BadRemapBlock;
         }
        goto Exit;
        break;
        case 3:
        SwapBlockAddr=FlashGetCurrentSwapBlock();
        for(i=0;i<(Addr&(FLASH_BLOCK_SIZE-1))/
                                    FLASH_PAGE_SIZE;i++)
         {
        if(0x00==(FlashCopyPage(SwapBlockAddr+
                    i*FLASH_PAGE_SIZE,RemapBlockAddr+
                                    i*FLASH_PAGE_SIZE)))
            goto BadRemapBlock;
         }
        if(0x00==(FlashCopyPage(Addr,RemapBlockAddr+
                                    i*FLASH_PAGE_SIZE)))
            goto BadRemapBlock;
        goto Exit;
        break;
        default:
        break;
    }
    BadRemapBlock:    FlashMarkRemapBlockBad
                                    (RemapBlockAddr);
}
    Exit:FlashUpdateBadBlockTable(Addr,RemapBlockAddr);
    return RemapBlockAddr+(Addr&(FLASH_BLOCK_SIZE-1));
}
```

3.5 连续读写操作

当主机与设备建立批量传输数据连接时，固件通过解析 CBW 封包获得起始 LBA。对该地址进行映射和坏块管理从而获得 PBA。设置一个变量，当此变量有效时，表示主机对 Flash 仍需要读或写数据，直到此变量失效为止。同时将主机上一次读写扇区的地址存入另一个变量，此变量在连续读中作用不大，但在连续写时，通过与上一次写操作地址所在块做比较，判断是否同属一个块，以此决定是不是需要进行地址跨块处理。

本文设计了一种针对 NAND 型的闪存转译层，使 NFTL 完成地址映射和坏块管理以及连续读写数据的操作。对 NAND Flash 的分区设计，使块管理结构清晰，有利于固件的开发。本文没有对 Flash 的 ECC 校验进行过多的设计，这是因为在实际应用中 NAND Flash 主要用于存储多媒体数据（图片、语音文件）等，并不会对它进行频繁的写入或擦除操作，而且多媒体文件数据对数据的完整性不敏感[5]，所以不需要对存储在其中的每一位数据进行严格的 ECC 校验，可以通过另外设计简单的校验方法来代替 ECC 校验。

参考文献

[1] 张雪，杨春林，黄娟.NAND Flash 文件系统的设计与实现[J].福建电脑，2007，24(10)：147-148.

[2] 罗晓，刘昊.一种基于 FAT 文件系统的 NAND Flash 坏块处理方法[J].电子器件，2008，31(2)：716-718.

[3] 郑桂芬.基于 USB 的可移动闪盘技术探讨[J].计算机工程，2003，29(7)：195-197.

[4] 刘荣.圈圈教你玩 USB[M].北京：北京航空航天大学出版社，2009.

[5] 杨玲，袁光涛.大容量 NAND Flash 在多媒体手机中的应用[J].科技咨询导报，2007，4(1)：22-24.

(收稿日期：2010-06-27)

作者简介：

侯超，女，1986 年生，研究生，主要研究方向：电路与系统。

徐建城，男，1986 年生，教授，硕士生导师，主要研究方向：电路与系统、微电子技术、无线传感网络。