[S3C2440的存储控制器](http://blog.csdn.net/hlyfx/article/details/8665738)

标签： [s3c2440](http://www.csdn.net/tag/s3c2440)[存储控制器](http://www.csdn.net/tag/%e5%ad%98%e5%82%a8%e6%8e%a7%e5%88%b6%e5%99%a8)[norflash](http://www.csdn.net/tag/norflash)[nandflash](http://www.csdn.net/tag/nandflash)

2013-03-12 21:53 1039人阅读 [评论](http://blog.csdn.net/hlyfx/article/details/8665738#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/hlyfx/article/details/8665738#report)

http://static.blog.csdn.net/images/category_icon.jpg 分类：

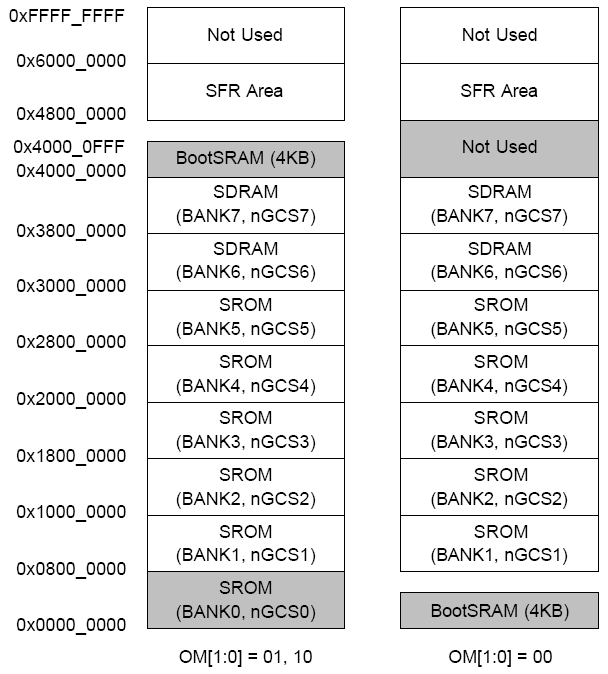
2440资源（1） http://static.blog.csdn.net/images/arrow_triangle%20_down.jpg

版权声明：本文为博主原创文章，未经博主允许不得转载。

**S3C2440存储控制器特性:**

1>.  s3c2440的存储器控制器为访问外部存储的需要提供了控制信号 （27位地址信号、32位数据信号、8个片选信号、以及读/写控制信号等）

查 S3C2440的手册可知S3C2440可寻址1G的地址范围，但是S3C2440的地址线只有27根，理论上只能寻址2的27次方等于128M的地址范围。那如何寻址1G的呢?S3C2440通过把存储管理系统分成8个Bank解决了这个问题。S3C2440引出了8根 Bank 线（对应nGCS0~ nGCS7）,通过这个8根线来选通和关闭不同的存储器，这样S3C2440最多就可以连接8个128M的存储器，只要在某一时刻只选通一个 Bank 就可以实现1G的寻址空间。每个 Bank 有个地址，对该 Bank 地址的访问实际上就是选通该Bank，于是ARM核只要发出一个地址，然后S3C2440的存储控制器只要把该地址解释成两部分：一部分是 Bank 地址，一部分是连接到该 Bank 存储器内部的地址就可以访问了。S3C2440是32位芯片，理论上讲可以达到4GB的寻址范围，除去上述8个 Bank 用于连接外部设备，还有一部分的地址空间是用于设备内部寄存器（GPIO,USB,PWM,LCD,RTC等寄存器），其余地址没有被使用。



2>.8个存储器Bank（ Bank 0— Bank 7），  Bank 0--- Bank 5为固定128MB， Bank 6和 Bank 7的容量可编程改变，可以是2、4、8、16、32、64、128MB, 最大共1GB

3>.Bank0可以作为引导ROM。其数据线宽只能是16位和32位，其它存储器的数据线宽可以是8位、16位和32位；

   Bank0~Bank5可以挂接ROM或SRAM类型存储器，Bank6~Bank7可以挂接ROM，SRAM，SDRAM类型存储器；

   Bank0~Bank5起始地址固定，bank7的开始地址与bank6的结束地址相连接（因为Bank6和Bank7是可编程访问的，故bank7的起始地址不      确定，但是二者的容量必须相等）；

    所有存储器bank的访问周期都是可编程的（可以8/16/32位，即可以以字/字节/双字为单位来访问Bank,假如挂载的是32为SDRAM，地址线的第二位A2与存储器的A0相连，这样存储器单元地址每次增加四，刚好对应相应的地指线A2增加1，同理，挂接的是8位的SRAM的话，则地址线A1和存储器的A0相连，挂载的是8位的话则地址线的A0和存储器的A0相连）;

   支持SDRAM的自刷新和掉电模式

//=====================================================================================//

注释：s3c2440的存储器控制器为访问外部存储的需要提供了控制信号。把存储系统分为8个bank（取名为bank而已），分别引出8根片选信号nCGS0~nCGS1，通过这八根片选信号来选通外部存储设备（即norflash，nandflash，SDRAM,ROM,RAM的片选信号引脚焊接在s3c2440的nCGS0~nCGS8上，其中Bank0~Bank5可以焊接ROM或SRAM类型存储器，Bank6~Bank7可以焊接ROM，SRAM，SDRAM类型存储器，也就是说，S3C2440的SDRAM内存应该焊接在Bank6~Bank7上，最大支持内存256M，Bank0~Bank5通常焊接一些用于引导系统启动小容量ROM，具体焊接什么样存储器，多大容量，根据每个开发板生产商不同而不同，比如MINI2440开发板将2M的Norflash焊接在了Bank0上，用于存放系统引导程序Bootloader，将两片32M，16Bit位宽SDRAM内存焊接在Bank6和Bank7上，并联形成64M，32位内存）

**2.Nor Flash和Nand Flash的区别和启动方式区别：**

1> NOR Flash 和Nandflash的区别: NOR Flash 和 NAND Flash 都是Flash的一种，但是 NOR Flash 价格相对较贵，读数据较快，写数据较慢，体积小，而 NAND Flash 价格比较便宜，读数据较慢，写数据较快，体积较大，一般 NOR Flash 用来存放bootloader， NAND Flash 用来存放内核[**操作系统**](http://lib.csdn.net/base/operatingsystem)和安装根文件系统。NOR Flash是总线型设备，可在芯片内执行(XIP，eXecute In Place)，应用程序可以直接在FIash闪存内运行，不必再把代码读到系统RAM中；而NAND Flash则需I／O接口，因此使用时需要写入驱动程序。

2>NOR Flash 和Nandflash的启动方式的区别：

----->Nor flash的有自己的地址线和数据线，可以采用类似于memory的随机访问方式，在nor flash上可以直接运行程序，所以nor flash可以直接用来做boot，采用nor flash启动的时候会把地址映射到0x00上

----->NandFlash没有接在bank0上，而程序必须从0地址开始执行，即bank0开始。2440内部有一个叫做“起步石（Steppingstone）”  的 SRAM缓冲器。系统启动时Nand flash存储器的前面4K字节被自动拷贝到Steppingstone中。Steppingstone被映射到nGCS0对应的BANK0存储空间。CPU在Steppingstone的4-KB内部缓冲器中开始执行引导代码。引导代码执行完毕后，自动跳转到SDRAM执行。

----->S3C2440的启动都是从0地址开始，所不同的是地址的映射不一样。在 S3C2440 开电的时候，要想让 S3C2440 知道以某种方式（地址映射方式）运行，不可能通过你写的某段程序控制，因为这时候你的程序还没启动，这时候 S3C2440 会通过引脚的电平来判断。

----->总结 1> 当引脚OM0跟OM1有一个是高电平时，这时地址0会映射到外部nGCS0片选的空间，也就是Norflash，程序就会从Norflash中启动， S3C2440 直接取Norflash中的指令运行(或者搬运代码到SDRAM中执行，nandflash是自动搬运代码的)。

                  2> 当OM0跟OM1都为低电平，则0地址内部bootbuf（一段4k的SRAM）开始。系统上电，S3C2440 会自动把NANDflash中的前4K内容拷贝到Steppingstone（内部SRAM缓冲器），并把0x00000000设置为内部SRAM的起始地址，cpu从内部SRAM的0x00000000开始启动，这个过程不需要程序干涉。（cpu会自动从NAND flash中读取前4KB的数据放置在片内SRAM里，同时把这段片内SRAM映射到nGCS0片选的空间（即0x00000000）。cpu是从0x00000000开始执行，也就是NAND flash里的前4KB内容。因为NAND FLASH连地址线都没有，不能直接把NAND映射到0x00000000，只好使用片内SRAM做一个载体。通过载体把nandflash中大代码复制到RAM(一般是SDRAM)中去执行。程序员要完成的工作是把最核心的代码放在nandflash的前4K中。这时NANDFlash中的前4K就是启动代码（他的功能就是初始化硬件然后在把NANDFlash 中的代码复制到RAM中，再把相应的指针指向该运行的地方）

mini2440就是把u-boot直接烧录在nor flash上。  nand flash是IO设备、数据、地址、控制线都是共用的,需要软件区控制读取时序,所以不能像nor flash、内存一样随机访问、不能EIP、片上运行,因此不能直接作为boot。 S3C2440把boot loader烧到nand flash上启动是因为在S3C2440里有一个内置的SRAM（叫做stepping stone(垫脚石很形象…)）系统启动加电后会把nand flash上的起始4KB的内容拷贝到SRAM里执行，这样就实现了从nand flash启动。如果bootloader小于4KB的话（像vboot）在SRAM里就能boot，大于4KB的话(u-boot、vivi)在SRAM里做一些基本初始化然后再把bootloader的剩余部分拷贝到SDRAM里（>0x30000000）。 //=====================================================================================//

注释：rbootloader是在操作系统内核运行之前运行的一段小程序。通过这段小程序我们可以初始化硬件设备、建立内存空间的映射图，从而将系统的软硬件环境带到一个合适的状态，以便为最终调用操作系统内核准备好正确的环境。通常bootloader是严重地依赖于硬件而实现的，特别是在嵌入式世界。因此在嵌入式世界里建立一个通用的bootloader几乎是不可能的。尽管如此，我们仍然可以对 Boot Loader 归纳出一些通用的概念来指导用户特定的 bootloader设计与实现。