文章编号:1671-4598(2019)10-0186-06 DOI:10.16526/j.cnki.11—4762/tp.2019.10.038 中图分类号:F407.63 文献标识码:A

基于分级缓存加速的高可靠高速星载固存设计

李 欣,禹霁阳,牛跃华,李宗凌,汪路元
(北京空间飞行器总体设计部,北京 100094)

摘要:星上模块在模式切换过程会因为星载固态存储器启动过程缓慢而导致无法快速访问固存;分级缓存系统设计采用片内缓存结合小容量非易失存储器(MRAM)的硬件架构,通过在MRAM中存储文件对象头索引以及Nand Flash块元数据区索引等流程优化来加速文件系统启动操作;该设计通过数据建模和仿真实验来分析性能,并在硬件板卡上进行算法实现和测试验证,以对象为索引的启动方式耗时3.12 ms,以块元数据区为依据的启动方式耗时143.47 ms;对比传统设计架构下的耗时170.35s的启动操作,基于分级缓存加速的系统具有高可靠性同时大大缩短了固存启动时间;其系统性能提升为卫星在轨管控优化提供技术基础。

关键词:高可靠;高速;分级缓存;星上数据管理系统

A High-Reliable Space-Borne Storage System with High-Speed Accelerated by Hierarchical Caching

Li Xin, Yu Jiyang, Niu Yuehua, Li Zonglin, Wang Luyuan
(Beijing Institute of Spacecraft System Engineering, Beijing 100094, China)

Abstract: During the mode switching process in the satellite, the storage system cannot be accessed quickly due to the slow initialization. The design of high-speed system based on hierarchical cache adopts the hardware architecture of on-chip cache combined with small capacity non-volatile memory (MRAM), in which the file object header index and Nand Flash block metadata area index are stored to accelerate the start-up operation of the file system. The performance of system design is analyzed by data modeling and simulation experiments. The system design is implemented and the start-up algorithm is verified in the on-board storage module. The start-up process based on object index takes 3.12 ms, and the initialization based on block metadata area takes 143.47 ms. Compared with the traditional start-up operation which takes 170.35s, this system design accelerated by hierarchical caching has high reliability and greatly shortens the start-up time of storage system. The improvement of its performance provides a technical basis for the optimization of satellite on-orbit management and control.

Keywords: high-reliability; high-speed; hierarchical caching; on-board data management system

0 引言

作为卫星数据管理系统中的关键组成,星载固态存储器为有效载荷业务、平台业务、星间中继通信业务等数据提供存储管理、回放传输等功能[1]。目前,星载固态存储器主要使用基于Nand FLASH的文件存储管理系统,其具有非易失性、电可擦除性、可重复编程以及高密度、低功耗等特点。但目前YAFFS法中的索引表使用多级映射方式,其索引表建立过程耗时巨大,从而导致星载固态存储器存在启动时间过长的问题。而卫星所处的空间环境充斥着大量的带电粒子会造成星载固态存储器运行逻辑错乱或者工作失效,严重时会导致整个星载数据管理系统不能工作。面对这种情况,地面通常会采用星载设备主备份切换或者复位操作来解决在轨出现的问题[2]。同时,空间探测器承载的有效载荷向着数量多,性能高以及多样化的方向迅速发展,卫星平台承载的数据流要求越来越高。面对高速的数据输入,固态存储器过长的启动时间会增加存储数据丢失或者出错的风险。因此如何减少星上存储系统的启动时间、提高快速访问速度已经成为当前星载数据管理研究的热点之一。

当前研究趋势主要集中于两个方向:1)通过精简索引表信息来减少需要恢复的数据量。例如Chan-Sul Park[3]提出的用扫描每个Nand Flash块(block)头文件空闲区的方式替代扫描每个Nand Flash页(Page)的方式来建立更精简的索引表信息。索引表的建立过程以及索引表内容均得到了简化,但其启动时间随着存储空间的增加而逐渐延长,不适用于大容量的Nand Flash存储系统;2)通过优化索引表存储方式来提高索引表建立速度。Keun Soo Yim[4]提出的快照技术(Snapshot)每次关机前将系统中的索引数据存储于Nand Flash中,在存储系统启动后通过读取快照中的数据来恢复索引表。但这种索引表存储方式减少了每次上电后对所有Nand Flash空闲区的扫描工作,成功加快了存储器启动速度。但其索引表的存储只在收到关机指令后启动,对与没有指令的非预期关机或者突然断电情况无法启动存储。而星上存储系统中常常存在这样非预期断电的情况,故该方式具有很高的不可靠性,不适于在卫星上使用。只包含Nand Flash单一存储器件的固态存储器,精

收稿日期:2019-03-26; **修回日期**:2019-04-17。

作者简介:李 欣(1985-),女,北京人,硕士,工程师,主要从事星上数据管理系统方向的研究。

简索引表信息常常受制于 Nand Flash 存储体的容量，而修改索引表的存储方式也会带来极大的不可靠性。

针对这一问题，分级缓存的新设计采用片内缓存结合小容量非易失存储器（MRAM）[5]的硬件架构打破这一困局。MRAM 其集动态 RAM、磁盘存储和高速缓冲存储器功能于一身，不仅存取速度快，掉电不流失数据，接口时序与 RAM 相似，而且存储单元不受单粒子效应的影响。本文将重点讨论如何将具有非易失且具有抗辐照功能的磁电存储器（MRAM）和 Nand Flash 存储器结合实现分级缓存，在以对象为索引的启动方式下把文件对象头存储指针存储在 MRAM 中，从而保证在启动时可以快速定位对象文件地址指针来建立文件系统。若恢复失败，再以 NAND FLASH 文件元数据为依据，重新精简元数据结构并集成为独立的块元数据区，对整个存储体的文件信息快速访问，从而实现固态存储器启动速度和存储可靠性的大幅度提升。

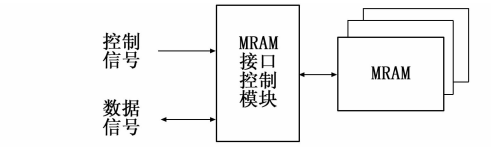
本文结构如下，第一节主要阐述了基于分级缓存加速的星载固态存储系统架构。第二节深入解析了基于分级缓存加速的文件系统启动及管理策略；第三节论述了星载固态存储器启动性能分析，并通过实验结果和以往设计进行了比较，表明了本设计的有效性；第四节对全文进行了总结和展望。

1　基于分级缓存加速的星载固态存储系统

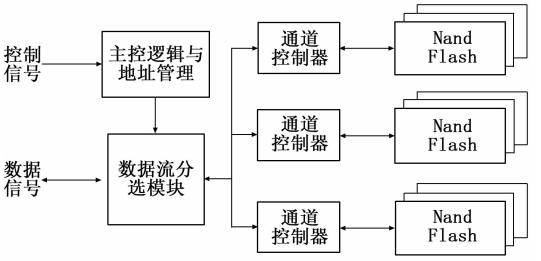
磁电存储器（MRAM）[5]中数据存储是通过直接附着于铁磁薄膜上具有电感耦合效应的导线来完成的。这种工作机理不仅提高了存储器的速度、可靠性，降低了功耗，而且在存储单元尺寸、存储速度方面也完全可以与 DRAM 相比拟。在卫星上使用的磁电存储器经过抗辐照加固和 3D 叠片封装，并行叠加的 MRAM 的数据线各自独立组成更宽的数据总线，共用地址线和读写控制信号，通过片选信号进行选通，如图 1（a）所示，例如 3D－plus 公司的 3DMR8M32VS8420 磁电存储器，其内部为两组 MRAM 存储基片，每次 4 片并行组成 32 位的数据总线，两组共用 17 位地址总线和读写使能信号，通过片选信号进行选通。

Nand Flash 芯片存储结构中的读写操作的基本单元为页（Page）[6]，擦除操作的基本单元为块（Block）。每页包含数据区和空闲区，在固态存储器中空余区用于存储该页相关的管理信息。由于 Nand Flash 的工艺问题，其只支持先擦除再写入的数据更新方式，此外其内部存在坏块，在坏块区存储数据会造成数据错误。Nand Flash 出厂时会标注存在的坏块，但坏块会在使用中陆续产生。在卫星上使用的 Nand Flash 存储器经过抗辐照加固和 3D 叠片封装。叠片中并行联系的 Flash 共用数据线，通过片选信号进行选通，可以使用一个 Flash 接口控制模块进行控制。而随着存储容量的增加，星上存储系统使用多个 3D 叠片封装的 Nand Flash 存储器，每个 Nand Flash 存储器的 IO 接口各自独立存在，从而形成 Nand Flash 的多通道存储架构[7]，如图 1（b）所示。

Nand Flash 和 MRAM 协同工作存储系统硬件架构，将



(a) MRAM存储架构



(b) Nand Flash存储架构

图 1　MRAM 和 Nand Flash 存储架构

MRAM 接入星载固态存储系统[8]的下层 FPGA 中。其和 Nand Flash 一起受 FPGA 控制，具体如图 2 所示。整个固态存储器系统按照层次化思想进行设计，其使用 CPU 作为固态存储器文件系统 YAFFS2 的实现载体，使用 FPGA 通过对 Flash 页读取，Flash 页写入、Flash 块擦除以及 Flash 坏块管理等基本单元操作，为 CPU 软件提供硬件接口支持。其中，FPGA 内部分为负责接口选通和时序的物理接口管理模块，负责存储单元操作的存储基本单元模块，负责数据流控制和纠错的数据缓存模块，以及和 CPU 总线通信的 CPU 接口模块。

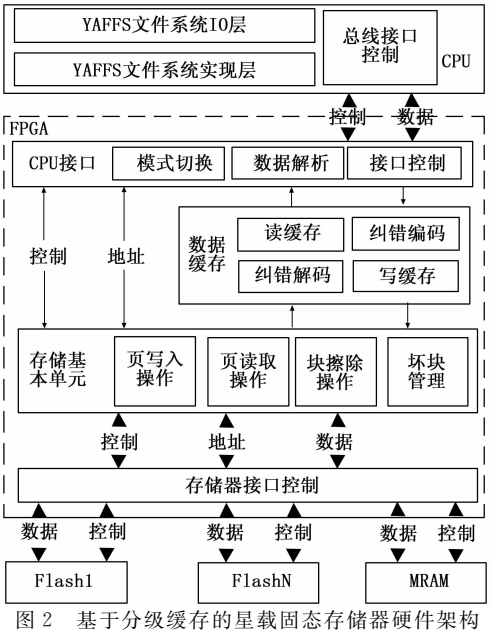


图 2　基于分级缓存的星载固态存储器硬件架构

2　基于分级缓存加速的文件系统启动优化设计

2.1　YAFFS2 文件系统启动流程

基于 YAFFS2 文件系统的固态存储器启动流程分为两种，一种为以对象为索引自上而下搭建文件系统，一种为以 Nand Flash 里面的元数据为依据自下而上的恢复文件系统，如图 3 所示。

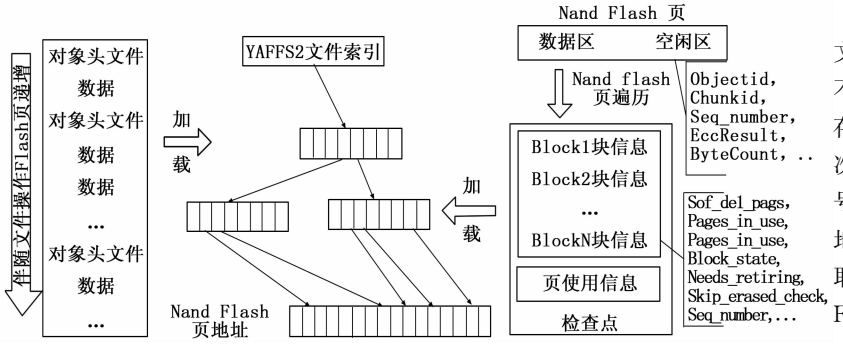


图 3 基于 YAFFS2 的固态存储器启动流程

第一种方式中，YAFFS2 使用对象（object）来实例化所有存入闪存的文件，每个对象都有一个对象头（object header）与之对应。对象头（object header）存储文件对象的基本信息，如对象类型、父目录的对象 id、对象名、属主信息、访问信息等等[9]。在文件系统挂载时，YAFFS2 根据对象头创建相应文件的对象，进而建立起整个文件系统的组织关系。每当文件发生改动（修改、增加、删除）后，文件系统都会在该文件后添加一个新的对象头做为标识，。由于文件有可能发生过多次更改，同一文件可能会对应多个对象头，但其中只有一个是有效的。在固态存储器启动时，需要扫描整个 Flash 来锁定有效的对象头文件，这使得文件系统加载时间随着 Flash 容量增大而延长，会出现启动速度慢的现象。

第二种方式中，YAFFS 按照固定大小的数据段来组织文件，并利用 NandFlash 页面的空闲区存放 ECC（错误检查与纠正）纠错信息和文件系统组织信息[10]。在系统启动过程中，固态存储器会遍历所有 Flash 页的空闲区来见获取足够的信息，将页信息整合为按照块为单位的块信息数组（block _ info）。遍历结束后，CPU 读取该数组到缓存中进行管理，并配置文件系统结构。同时 CPU 在内存中设置 chunk _ bits 数组记录 Flash 中所有逻辑页的使用情况，数组中的每一个位（bit）对应闪存中的一个页的状态，若页被使用则对应位设置为 1。当文件系统被卸载时，块信息数组（block _ info）和 chunk _ bits 数组作为检查点（checkpoint）数据被写入到 Nand Flash 中[11]。在下一次加载文件系统时，检查点数据可以帮助 YAFFS2 将文件系统快速恢复到上一次卸载前的状态，而不必扫描整个 Flash，从而加快了文件系统的挂载速度。如果文件恢复失败，YAFFS2 读取块序号（seq _ number）采用逆序扫描的方式获取闪存上每一块的信息来恢复文件系统。这样的遍历操作也会造成启动时必须遍历所有块，致使启动时间随容量增加大幅增长，无法满足星载固态存储器的应用需求。

2.2 基于分级缓存加速的启动及管理策略

通过对启动流程进行分析可知，第一个从对象出发的启动方式的关键在于对各个文件中对象头（object header）的定位。第二个从 Flash 页内空闲区出发的启动方式的关键在于对索引信息的维护。因此，启动流程的优化关键在于文件对象头的地址索引，以及块信息索引表的数据维护。

新的对象头页面（Object header）伴随着文件系统操作而不断被创建，旧的对象头页面不断失效。本文将这类灵活变化又需要断电保存的地址信息存入磁电存储器 MRAM 中，每次新对象头页面被创建时就同步更新对应文件号（ObjectID）在 MRAM 中的对象头页面存储地址，这样固态存储器启动的时候可以直接读取最新对象头文件的存储指针，减少了遍历 Flash 的操作流程。

针对块信息索引表的数据维护，本文旨在通过精简文件组织信息内容和优化文件组织信息存储方式等两个方面实现启动流程的优化。在文件组织信息中，Nand Flash 存储体的坏块信息更新频率低，查询率高，适合在吞吐高，断电保存的 MRAM 中存储。因此在本设计中，在 MRAM 中设置坏块表记录 Flash 中所有块的好坏块情况，表中的每一个位（bit）对应 Flash 中的一个块，好块为 0，坏块为 1。通过对好坏块表的查询可以成功屏蔽坏块对文件操作的影响。

参考 yaffs _ PackdTags2Part 描述结构内容，提炼文件信息数据，即块元数据。考虑星载系统处理数据的类型与特点，设置星载固态存储器的文件系统最多支持 1024 个文件，同时每个文件的最小单位为一个 Flash 的块。这样对于每个 Flash 的块来说，其元数据中的每个属性都是唯一的，其具体参数如表 1 所示。

表 1 Nand Flash 块元数据内容列表

偏移地址/bits	名称	大小/bits	备注
0－31	块序号	32	当新块被分配并存入数据时即获得一个块序号，该序号依次递增。
32－35	块属性	4	支持 10 种运行状态。
36－47	文件号	12	所在块支持的文件号。
48－55	文件占用页数	8	支持最大 256 页的块
56－63	ECC 校验	8	ECC 校验码可以自动纠正 1 位错误，被用于对抗空间中的单粒子效应带来的数据错误。

本设计尝试在 Nand Flash 中寻找区域集中保存所有的块元数据，定义为块元数据区，特别针对星载固态存储器的多通道系统，在每个拥有独立 IO 接口的 Flash 芯片（Device）中设置保存该 Device 所有 Block 块元数据的块元数据区，这样在固态存储器启动时可以并行多通道读取各自独立的块元数据区，来获得整个 Nand Flash 存储体的文件系统信息，而不再将每个 Block 块都一一遍历，从而大大缩短启动时间。其具体存储分布和更新方式如图 4 所示。

伴随着星载固态存储系统的运行，每个文件操作都有可能导致 Block 块元数据内容的失效，故需要定时遍历整个 Device 来读取 Block 块内部文件信息形成新的块元数据区文件。从旧的块元数据区文件尾部开始顺序存储到 Flash 中。最新一次的完整数据的地址指针伴随着每次定时更新而跳

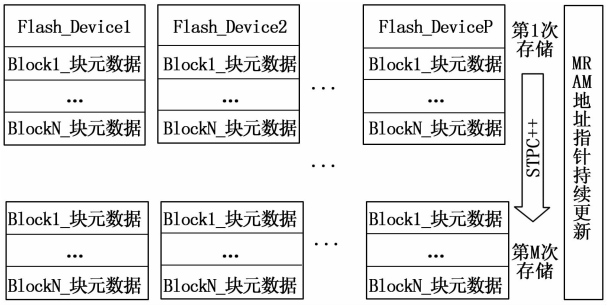


图 4　多个 Device 中块数据区存储及更新示意图

变，故将这部分数据放入 MRAM 中存储。当断电后启动时，星载固态存储器来读取 MRAM 来获得最新一次版本的块数据区的完整数据，并作为数据恢复的信息参考。将整个存储体的坏块表，所有文件的对象头存储指针以及所有 Flash 芯片的块元数据区存储在 MRAM 中进行管理，通过三模冗余存储设计来保证存储数据的高可靠性。

3　星载固态存储系统启动性能分析

3.1　基本假设

星载固态存储器的启动时间与 Nand Flash 接口时序、容量、内部坏块数、块元数据区大小以及块元数据更新周期等因素有关，依照星上器件使用情况对上述因素进行量化假设：

1）Nand Flash 存储体内部架构以及相关参数在目前星上存储系统的性能区间如表 2 所示。

表 2　星载固态存储器中 Nand Flash 内部参数表

参数字符	参数定义	星上应用数值范围
B_{Data}	每页 Page 的数据区字节数	$2048 \leqslant B_{Data} \leqslant 8192$
B_{SPARE}	每页 Page 的空闲区字节数	$64 \leqslant B_{Spare} \leqslant 224$
Q_{Page}	每个 Block 块内部包含 Page 页数	$64 \leqslant Q_{Page} \leqslant 128$
S_{Block}	每个基片 Chip 中包含 Block 块数	$2048 \leqslant S_{Block} \leqslant 4096$
L_{Chip}	每个 Flash 芯片内部叠加的基片数	$4 \leqslant L_{Chip} \leqslant 8$
P_{Device}	固态存储系统中装配 Flash 芯片个数	$1 \leqslant P_{Device} \leqslant 256$

2）Nand Flash 读取操作时，设定时钟周期为 t_{RC}，Flash 内部运行时间为 t_{rb}，读取数据字节数为 B_{rd}，Flash 内部数据的读取时间为：

$$T_{rd} = 7t_{RC} + t_{rb} + B_{rd}t_{RC} \tag{1}$$

设置 $T_{pre} = 7t_{RC} + t_{rb}$，则方程式（1）可以简化为：

$$T_{rd} = T_{pre} + B_{rd}t_{RC} \tag{2}$$

其中：t_{RC} 一般在 40～80 ns 之间，t_{RC} 一般在 25 μs 左右。

3）MRAM 读取时间为 T_{M_RD}，其中 $0 \leqslant T_{M\_RD} \leqslant 35$ ns

4）星载固态存储器 YAFFS2 文件系统中文件数设置为 W_{Object} 个，其中 $0 \leqslant W_{Object} \leqslant 1023$。

5）单个 Device 中块元数据区大小为 B_{WA} 个 Flash 数据页 Page，根据定义，即：

$$B_{WA} = \frac{8L_{Chip}S_{Block}}{B_{Data}}$$

设定 $0 \leqslant B_{WA} \leqslant Q_{Page}$。

3.2　启动时间建模

根据上一节 Nand Flash 读取时间公式（2），对传统算法的启动过程进行建模。传统星载固态存储器每次启动需要搜索所有数据页的空闲区，来寻找对象头文件，或者获取块序号（Seq _ Number）用于逆序扫描建立文件系统。因此其启动时间 T_{mnt} 可以定义为：

$$T_{mnt} = P_{Device}L_{Chip}S_{Block}Q_{Page}(T_{pre} + B_{Spare}t_{RC}) \tag{3}$$

本文中设计的星载固态存储器针对两种 YAFFS2 文件系统挂载方式分别设计了基于 MRAM 的启动加速算法，下面一一进行建模并估算启动时间：

1）针对以对象为索引自上而下的文件启动方式，其启动过程为读取 MRAM 获得对象头文件地址指针，然后读取头文件内部数据，依次遍历所有文件后，从而完成固态存储器启动操作。故其启动过程 T_{mnt_H} 可以定义为：

$$T_{mnt\_H} = W_{Object}[T_{M\_RD} + T_{pre} + (B_{Spare} + B_{Data})t_{RC}] \tag{4}$$

考虑到 T_{M_RD} 远小于 T_{pre}，则公式（4）可以简化为：

$$T_{mnt\_H} = W_{Object}[T_{pre} + (B_{Spare} + B_{Data})t_{RC}] \tag{5}$$

2）针对以 Nand Flash 里面的元数据为依据自下而上的文件启动方式，其启动过程为读取 MRAM 获得每个 Device 中的块元数据区起始地址指针，然后数据区的块元数据，从而完成文件系统加载操作。在这个过程中，读取 MRAM 的操作室串行的，而读取各个独立 Flash 的操作则是并行进行的，故其启动过程 T_{mnt_D} 可以定义为：

$$T_{mnt\_D} = P_{Device}T_{M\_RD} + B_{WA}(T_{pre} + B_{Data}t_{RC}) \tag{6}$$

引入公式（3），并考虑到 $P_{Device}T_{M\_RD}$ 数值太小可以忽略，公式（5）可以演变为：

$$T_{mnt\_D} = 8L_{Chip}S_{Block}\left(t_{RC} + \frac{T_{pre}}{B_{Data}}\right) \tag{7}$$

通过公式（5）可知，针对以对象为索引自上而下的文件启动方式只受文件数量和单页 Flash 读取时间影响，而 Nand Flash 芯片容量和内部结构不再直接影响启动时间。参考 3.1 节中各个参数的数值范围，当文件数量为 1024，单页 Flash 数据区字节数为 8192，空闲区字节数为 224，Flash 读取时钟周期为 80 ns 时，理论上按照公式（5）获得最大启动时间为 689.28 ms。

通过公式（7）可知，针对以 Nand Flash 里面的元数据为依据自下而上的文件启动方式受单个芯片 Device 的内部块结构和单字节 Flash 读取时间影响，而与 Nand Flash 芯片个数关系不大。参考 3.1 节中各个参数的数值范围，当单页 Flash 数据区字节数为 2048，单个基片 Chip 中包含 4096 块，每个芯片包含 8 个基片，Flash 读取时钟周期为 80 ns 时，理论上按照公式（7）获得最大启动时间为 24.17 ms。

通过公式（3）可知，以传统方式启动，启动时间与存储器的整体规模直接相关，存储体容量越大，时间越长。参考 3.1 节中各个参数的数值范围，当单页 Flash 空闲区字节数为 64，存储体包含 1 个 Nand Flash 存储芯片，单个存储芯片中包含 4 个基片，单个基片 Chip 中包含 2048 块，单

个 Block 块内部包含 64 页，Flash 读取时钟周期为 40 ns 时，理论上按照公式（3）获得最小启动时间为 22.84 s。

由此可见，针对星载固态存储器的使用区间，经过分级缓存加速优化的启动流程的最大耗时都远远小于基于传统设计方案的启动方式的最小耗时。故针对星载固存设计来说，分级缓存加速的文件系统启动速度大大优于传统文件系统启动速度。

3.3 实验结果

本次设计中，FPGA 芯片选用 Xilinx 公司的 XQR2V3000，设置外部激励对固态存储器上电启动进行控制仿真。通过仿真，基于硬件加速的星载固态存储器启动设计的性能可以得到更多方面的论证。

1）设置文件对象数为 1000，每个芯片 Device 包含 4 个基片，每个 Flash 基片（Chip）包含 4096 的块（Block），Nand Flash 接口读取时钟为 60 ns，选取不同规格的 Nand Flash 进行仿真，测试 Flash 单片容量与启动时间的关系，具体如图 5 所示。

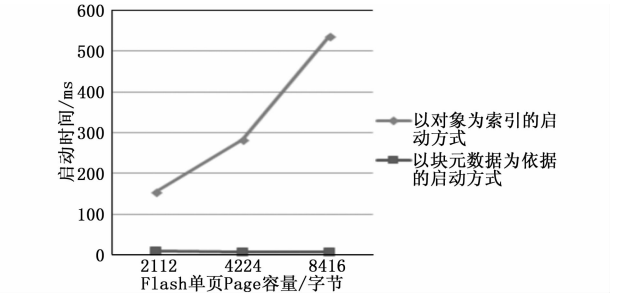


图 5 不同规格 Nand Flash 环境下固态存储器的启动时间

图表中的具体数值如表 3 所示。

表 3 单片容量不同的 Nand Flash 的启动时间

单页字节数	以对象为索引的启动时间/ms	以 Nand Flash 里面的元数据为依据的启动时间/ms
2112	155.1	9.45
4224	284.3	8.75
8416	536.9	8.23

通过图 5 的仿真结果可以看出，当单片 Page 容量很小的时候，两种方式性能差不多。随着 Nand Flash 单片容量的增加，星载固态存储器的启动时间因为采用启动方式的不同而出现截然不同的特性。在以对象为索引的文件启动方式下，启动时间伴随容量增加而明显延长。而在以 Nand Flash 里面的元数据为依据的文件启动方式下，启动时间反而逐渐减小并趋于一个稳定值。也就是说，当单片 Page 容量超过 8 416 字节后，启动时间不再受单片 Page 大小的影响。

2）设置文件对象数为 1 000，每个芯片 Device 包含 4 个基片，每个 Flash 基片（Chip）包含 4 096 的块（Block），每页 Page 数据区包含 4 096 字节，空闲区包含 128 字节，通过 FPGA 修改 Nand Flash 读取时钟周期（t_{RC}），测试 Flash 读取速率与启动时间的关系，具体实验结果如图 6 所示。

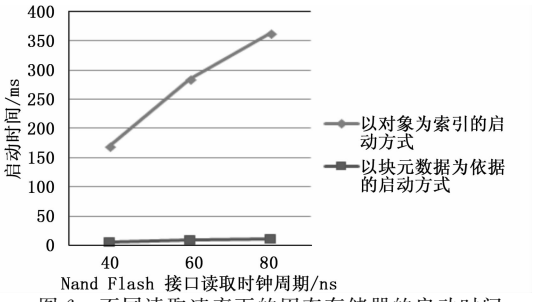


图 6 不同读取速率下的固态存储器的启动时间

图表中的具体数值如表 4 所示。

表 4 不同读取速率下固态存储器的启动时间

读取时钟周期/ns	以对象为索引的启动时间/ms	以 Nand Flash 里面的元数据为依据的启动时间/ms
40	170.1	6.28
60	284.3	8.75
80	362.9	11.36

通过对实验数据进行解析发现在两种启动方式下，星载固态存储器都不同程度地随 Nand Flash 接口读取速率的提高获得更高的启动速率，但星载设备的可靠性要求接口时序拥有一定的时序裕度，故要求使用者根据具体应用环境选择合适的 Nand Flash 接口读取时钟周期，从而在高启动速率和高可靠性之间得到平衡。

3.4 与传统设计进行对比

在实际的测试板上面，选用 Xilinx 公司的 XQR2V3000 作为芯片载体，选用五块三星公司出品的 FLASH 存储器 MMFN08408808S－F 和一块 3D_Plus 公司出品的 MRAM 存储器 3DMR8M32VS8420 作为存储器件，使用 YAFFS2 文件系统，创建 10 个文件，进行断电重启操作，记录上电后到文件系统组织信息搜集完毕的时间。经过测试，以对象为索引的文件启动方式获得的启动时间为 3.12 ms，以 Nand Flash 里面的元数据为依据的文件启动方式获得的启动时间为 143.47 ms。在相同的实验条件下，以传统设计方式进行重启后，其启动时间为 170.35 s。由此可知，本文通过加入 MRAM 实现分级缓存的硬件架构并设计了基于分级缓存文件系统的启动加速算法，这样的设计可以大大缩短了星载固态存储器启动时间。

4 结论

本文从星载固态存储器在轨工作环境出现的问题出发，深入研究星载固态存储器传统启动策略的优化局限，提出一种基于分级缓存加速的星载固态存储器启动及管理算法。分级缓存设计在硬件架构上引入 MRAM 磁电存储器来存储 Nand Flash 的坏块信息和文件系统操作中容易变化文件组织信息。针对 YAFFS2 文件系统的加载过程，分级缓存对关键信息的高吞吐率保证固态存储器直接定位对象头文件而缩短了基于对象索引建立文件系统的时间，也可以通过直接定位更精简集成的块元数据区的起始地址来加速以 Flash 元数据恢复文件系统的进程。MRAM 的抗辐照以及

断电保护特性，保护其内部存储文件数据的正确性和实时性，大大提高文件系统可靠性。在本文中，作者对两种启动方式下的优化设计进行建模，通过性能分析和与传统算法的比较，以及在星载存储测试板上的实验验证均表明新设计的启动时间大大缩短，从而证明基于分级缓存加速的固态存储器启动设计拥有超越传统设计的优越性能。

参考文献：

[1] Mcmickell M B，Tanzillo P，Kreider T，et al. Rapid development of space applications with responsive digital electronics board and LabVIEW FPGA [A]. NASA/SA Conference on Adaptive Hardware and Systems (AHS) [C]. Anaheim：[s. n.]，2010：79－81.

[2] 李　姗，宋　琪，朱　岩，等. 星载大容量固态存储器快速可靠启动算法设计 [J]. 哈尔滨工业大学学报，2015，10，47 (10)：116－121.

[3] Park C S，Han T H. Fast mounting method for NAND flash memory file system using offset information [A]. International Conference on Advanced Communication Technology (ICACT 2010) [C]. 2010：675－679.

[4] Yim K S. A fast start－up technique for flash memory based computing systems [A]. Proceedings of the ACM Symposium on Applied Computing (SAC) [C]. New York：ACM，2005：843－849.

[5] Chen Y S，Wang D Y，Hsin Y C，et al. On the hardware implementation of MRAM physically unclonable function [J]. IEEE Transactions on Electron Devices，2017，64 (11)：4492－4495

[6] Kim C G，Kim K J，Lee J. NAND flash memory system based on the Harvard buffer architecture for multimedia applications [J]. Multimedia Tools and Applications，2015，74 (16)：6287－6302.

[7] Kwak J，Kim H C，Park I H，et al. Anti－forensic deletion scheme for flash storage systems [A]. Network Infrastructure and Digital Content (IC－NIDC) [C]. 2016：317－321.

[8] 徐　勇，曾连连. 面向 FPSS 的通用星载 NandFlash 存储器设计 [J]. 飞行器测控学报，2012，12，31：60－64.

[9] Schierl A，Schellhorn G，Haneberg D，et al. Abstract specification of the ubifs file system for flash memory [A]. FM 2009：Formal Methods [C]. Springer Berlin Heidelberg，2009：190－206.

[10] Qian Y Q，Lu J J，Xing K. Wear－Leveling Optimization of Android YAFFS2 File System for NAND Based Embedded Devices [J]. Lecture Notes in Computer Science，2014，8491 (1)：12－21.

[11] Chung T S，Park D J，Kim J. An efficient flash translation layer for large block NAND flash devices [J]. Journal of Circuits，Systems and Computers，2015，24 (09).

(上接第 170 页)

障预案模型的传导消耗时间。

经过两套设备对遥测信号的采集和运算，得到如下的性能对比。

表 1　两套设备性能对比　　ms

方案名称	传统方案	实时模型方案
解算延时均值	3000	1000
解算延时误差	200	5
故障预警同步延时	500	1

5　结论

本文首先分析了当前卫星综合测试系统的发展前景和面临的挑战，从遥测信号的重要性和特性出发，着眼于新的基于实时模型遥测信号的解析方案。

首先从成本控制的角度出发，基于实时模型遥测信号的解析方案压缩了原有的硬件成本，将传统方案中的至少两台套的设备功能集成在一台设备中进行实现。

其次，从性能上分析，解算延时均值、解算延时误差和故障预警同步延时等指标大幅提高，提升了故障预案的综合性能。

综上所述，本文通过采用实时系统模型架构和简单的模型模块配置，便可实现多种遥测信号采集以及完成与故障预警模型数据交换，达到了快速建模、分析和仿真的目的，可大幅提升卫星综合测试系统性能。

参考文献：

[1] 孙泽洲. 通用化卫星测控地面测试系统设计 [J]. 航天器工程，1999，8 (1)：37－41.

[2] 傅海燕. 卫星遥测数据的时间序列相似性度量方法研究 [D]. 哈尔滨：哈尔滨工业大学，2015.

[3] 杜　楚，彭会湘，陈　勇. 以文档结构存储海量卫星遥测数据 [J]. 无线电工程，2017，47 (1)：46－48，75.

[4] 吴　婧，苏振华，孙　诚. 一种基于 Kalman 滤波的卫星遥测数据判读系统 [J]. 航天器工程，2014，23 (3)：86－91.

[5] 邓昌义，郭锐锋，张忆文，等. 面向硬实时系统零星任务低调度算法 [J]. 小型微型计算机系统，2016，37 (1)：157－161.

[6] 张庆华，吴　印，韩吉韬. 基于通用接口母线的自动测试方法 [J]. 南京理工大学学报 (自然科学版)，2005，29 (1)：35－39.

[7] 张俊娇. 比特流协议分析与特征识别技术研究 [D]. 成都：电子科技大学，2016.

[8] 张黎明，孙　宁，于慧亮，等. 基于 PXI 的卫星综合测试系统的设计与实现 [J]. 计算机测量与控制，2008，16 (1)：27－29.

[9] 孟祥迪，郭静寰，熊木地. 基于 PXI 总线技术的星载计算机性能测试系统研究 [J]. 计算机测量与控制，2007，15 (5)：571－573.

[10] 何铭俊，洪　雷，曹丽君，等. 一体化小卫星综合测试系统的设计 [J]. 计算机测量与控制，2016，24 (9)：15－18.