# TI c66x 系列 DSP 多核 BOOT 的研究

钱丰, 林家儒

(北京邮电大学信息与通信工程学院信号与信息处理实验室, 北京 100876)

摘要: DSP发展到 21 世纪初期,其架构和规模不断的扩大,启动方式也朝着多元化和自动化的角度发展。其内置 ROM 功能不断的强化和升级,保证用户只要深刻理解 Boot 原理并处理好镜像与 Boot ROM 接口关系就可以方便实现各种方式的加载。多核 DSP 加载处理的流程较为复杂,涉及到二级引导和核间中断的问题。针对于此文章基于评估板 c6678EVM对 8 核 DSP 的加载做研究分析和局部优化。优化后的方法解决了 TI 工具链隐藏的一些兼容性问题同时使得多核启动变得更加方便快捷。

关键词: 数字信号处理器; 核间中断; 引导程序; 多核启动

中图分类号: TN911.72

5

10

15

20

25

30

35

40

# Multicore Boot rearch on TI c66x DSP

Qian Feng, Lin Jiaru

(ITTC Laboratory of Beijing University of Posts and Telecommunications, Beijing 100876) **Abstract:** As DSP develops to the early 21st century, its architecture and scale has changed a lot. The corresponding Boot Method is also progressing towards the development of diversified automation point of view. Its built-in ROM with continuously enhanced and upgraded function ensure that the principle of the user is defined as deep understanding of the Boot, and well designing of the interface with the Boot ROM so that a variety of load can be achieved. Multi-core DSP's loading procedure is much more complex than that of single-core DSP 's. The former involves second-level boot program and inter-core interrupts. This article is based on the c6678EVM board designing the loading of the 8-core DSP and local optimization. The optimized method is to solve some compatibility problems which TI tool-chain hides and to make start multi-core DSP more convenient.

Keywords: DSP; Inter-core interrupt; Boot program; Multicore boot

# 0 引言

自 2010 年 TI 公司推出 keystone c66x 系列的 DSP 之后,关于该种多核 DSP 启动方式的研究也成为技术难点之一。新型架构的 DSP 由于其结构的复杂性和较高的灵活性等特点,能够允许多渠道的加载方式:本地的或远程的,基于 ROM 的或基于 flash 的。总结起来一共有以下 7 种:EMIF16 启动、SRIO 启动,以太网启动、PCI-e 启动、主从  $I^2C$  启动、SPI 启动以及 Hyper-Link 启动。

从软件固化的思想考虑,绝大多数启动方式都将代码放入 EEPROM 或者 flash 作为启动镜像。但由于 EEPROM 容量小,造价高,所以  $I^2C$  启动模式主要应用于代码量较小的 Demo 板或其他测试板卡中。最常用的基于 flash 的加载方式是通过 EMIF 口或 SPI 接口进行 Nor 、Nand flash 的烧写进行启动。

文章主要介绍了 c66x 系列 DSP 基于 Nor flash 的 SPI 接口的加载原理,分析了其工具链的使用限制和优化方法。并最终给出一个方便易行的加载方案。该方案很好地解决了 c66x 的多核加载。

\_

**作者简介**: 钱丰,(1988-), 男, 硕士研究生,基于 TI c6000 DSP 的架构设计和代码优化。 **通信联系人**: 林家儒,(1958-),男,教授,主要研究方向: 信息与通信理论、移动通信、通信系统等。 E-mail: jrlin@bupt.edu.cn

http://www.paper.edu.cn

# 1 多核 DSP 启动

### 1.1 典型 c6000 的加载

45

50

55

65

70

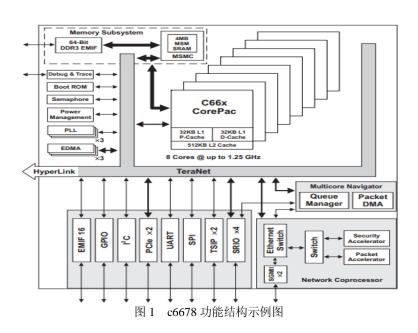
在 ROM 或 flash 中的数据断电后不会丢失,嵌入式产品中的程序一般都是存在这两种介质当中。 ROM boot DSP 的模式下,C621x/C671x/C64x 系列的 DSP EDMA(外部直接存储器存取) 会自动的从 CE1 存贮器映射空间 (对 6416 是 0x6400 0000) 拷贝 1K byte 的代码到地址 0x00000000 (在 DSP 内部的 SRAM),其间 CPU 被停止,然后 CPU 才从地址 0 开始运行。

这样,如果应用程序的机器码小于 1k bytes,我们可以把这些机器码烧写进 flash 中的前 1k bytes(此 flash 必须接到地址空间的 CE1,对 641x 系列是地址 0x6400 0000), DSP 一复位,程序就自动由 EDMA 加载进 DSP 内部 SRAM,开始执行。这个加载过程叫做 First Level bootloader (一级加载)。读 flash ROM 的操作完全是硬件自动控制的,时序是默认的时序,如果 CPU 跑到 600MHz, EMIF-B (外部存储器接口 B) 时钟选六分频,读 1k bytes 大约用 0.83ms。

如果程序的机器码大于 1k bytes, , 不妨假定应用程序的主体代码(>1M bytes)已经被烧写在 CE1 上的 FLASH 之中, 而且没有占用 CE1 的起始 1k bytes, 利用这 1k bytes 写个简单的汇编程序, 把应用程序的主体代码从 FLASH 中拷贝到 DSP 的内部 SRAM, 然后再跳转到 C 语言环境初始化程序的入口处 "\_c\_int00", 启动的任务就可以完成。这 1 K byte 的程序就叫做 Second Lever Bootloader (第二级启动加载程序)。

## 60 **1.2 C66x** 的改进

C66x DSP 内部有一个固化的 ROM,里面存放着 boot 代码(ROM bootloader)。每当 DSP 启动时,



会自动从这里读取代码并执行。这里执行的代码是固化的不可更改的,其作用就是根据 DSP 的管脚配置方式对核进行初始化(比如 PLL 等)和完成不同模式的 Boot 处理。所有的 core 执行同一份代码。不同的 core 在执行的时候通过 DNUM(核编号索引)来去做区分。 初始化外设的操作由 core0 来完成。所以 core0 初始化其他外设的同时,其他 core 都会执行

相关的代码映射 IPC 中断,并配置相应的寄存器,然后进入 IDLE 状态,等待 core0 的 IPC 中断发起。简而言之,其他核是在 core0 的命令下执行第一句代码。

与之前 c6000 系列不同的是,c66x 系列 boot loader 不再局限于 1k 字节代码的搬运。用一定的工具链将每个核的代码按照指定次序存放,ROM boot loader 甚至可以直接实现 8 个 core 的代码搬运和加载而无需用户的干预。

# 1.3 Boot-loader 的初始化

75

80

85

90

95

100

电源开启后 DSP 板卡冷启动,这意味着所有的外设都进入默认状态同时开始 boot 的初始化工作。初始化所需要的配置信息全部从设备状态寄存器里(DEVSTAT)读取,该寄存器值会通过扫描一个 13 管脚的拨码开关来确定其默认值。读取配置信息后 Bootloader 会继续相应的 boot 操作。但是无论进行各种方式的启动,一定会执行以下步骤:

- 固化在 ROM 上的代码段会让所有支持重启隔离功能的外设使能重启隔离功能。
- 固化在 ROM 上的代码段会使能 boot 过程中所有可能需要用到的外设的时钟。
- 固化在 ROM 上的代码段会利用从 DEVSTAT 寄存器三个 PLL 比特中读取的信息 配置系统 PLL。
- 如果以无自启动、SPI 启动或者 I2C 方式启动, 主 PLL 会被配置为旁路模式; 若以 其他方式启动, 固化在 ROM 上的代码会以 PLL 模式去配置主 PLL。
- 固化在 ROM 上的代码段会初始化 core0 的最后 0xd23f 个字长的代码, 此段代码用于保存 boot 配置信息亦即 boot 参数表。

偏移 长度 描述   0x0000 0x0040 ROM Boot 版本   0x0040 0x0400 Boot 代码堆栈	表 1 ROM bootloader 往核 0 中的內存分配						
0x0040 0x0400 Boot 代码堆栈							
0.0500 0.0000 - )!! 45 5 5 55 55							
0x0520   0x0020   Boot 进程寄存器堆料	浅						
0x0540 0x0100 Boot 内部状态							
0x0640 0x0100 Boot 变量							
0x0740 0x0100 DDR 配置表							
0x0840 0x0080 RAM 表函数							
0x08c0							
0x0940 0x3600 清除明文漏洞							
0x5240 0x7f80 网口、SRIO 口、包、指	描述符						
0xd130 0x0080 小堆栈							
0xd23c 0x0004 Boot Magic 地址	Boot Magic 地址						

表 1 ROM bootloader 在核 0 中的内存分配

### 1.4 Boot Magic 地址

Boot Magic 地址是每个 core 各自一块固定的内存。由上表所示,Boot Magic 地址是ROM 搬移到RAM 信息的最后一个字。该字存放的是各个 core 初始化之后需要跳转到的 c程序入口地址 \_c\_int00()。根据 c66x 内存的规划设计,不同 core 的 Boot Magic Address 存位于在该 core 本地 L2 RAM 的最后一个 word 里。由于多核 DSP 采用全局地址来区别不同核的 RAM 地址,因此每个核的 Boot Magic 地址是 0x1x87fffc(x 为核号)。

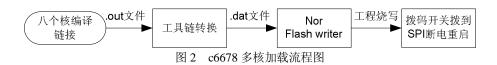
当 ROM Boot loader 写完 Boot Magic 地址之后, core0 作为主核会发起一个 IPC 中断给 其他核用于唤醒处于 IDLE 状态的从核,并同时将各核各自的 Boot Magic 地址写入 0x1x87fffc。从核会读取该 Boot Magic 地址,并从该地址处执行程序。

# 基于 Nor FLASH 的多核 DSP 启动

SPI Boot 是通过 SPI 接口从挂载 Nor flash 里加载程序进入 DSP 内存的过程。以 EVM6678LE 评估板为例,上面挂载的 Nor flash 大小为 16Mbytes,可以提供足够大的程序 烧写空间。烧写过程可以利用 TI 提供的工具 Nor-Flash writer。

#### 2.1 基本流程 105

下图阐释了通过 Nor flash 烧写加载程序的全部流程:



110

115

120

125

130

135

基于 CCSv5 IDE 下,八个核独立编译并各自生成属于每个核的.out 文件。由于携带了 大量的调试信息,.out 文件一般不直接烧写到 flash 上,而是要经过一些工具链的转换进行 筛选,将有用的信息保留。同时由于涉及到多核,还需要将每个核的有用信息按照一定的顺 序链接起来,一次性加载进入 flash,而不是像之前 c6000 各系列芯片上的启动方式,需要 靠用户自定义引导程序加载多核代码。

工具链的另外作用是为镜像加 Boot 参数头。DSP 内部的 ROM bootloader 需要知道当 前启动的一些信息,包括启动方式、大小端模式、PLL 频率等参数。将这些信息做成一个参 数头列于八个核各镜像信息的前面才构成了一套完整的多核加载文件。文件的烧写可以使用 TI 提供的 MCSDK 下的 Nor-Writer 程序。由于通过 DDR 暂存镜像文件, 烧写程序在初始化 EVM 板的时候需要选择 no-boot 模式, 只有在这种模式下 gel 文件才会完成 DDR3 的初始化 操作。

烧写完毕后,拨码开关需要重新拨成 SPI boot 模式。这时候重新启动,以走马灯程序为 例, EVM 板的 LED 灯会被成功点亮。8 个核被启动成功。core0 在 SPI 启动中起到至关重 要的作用。作为主核,核 0 负责着搬移镜像,写 Boot Magic 地址值,以及发 IPC 中断触发 其他核工作的多项工作。其中搬移多核镜像的工作由 DSP ROM Bootloader 完成,而剩下两 项工作需要用户自定义操作。



如果多核 DSP 是由同一套工程分别编译,那么每个核内存分配完全相同。core0 在读取 自己核的 Boot Magic 地址(0x1087fffc)后,加上 0x0\*000000 后就可以得到其他核的 Boot Magic 地址(\*为核号)。但是如果各核编译各自独立工程,各变量内存映射关系不再相同, 那么就无法从 core0 的 Boot Magic 地址里的值去推算其他核相应地址。这个时候只能事先记 录下各核的 Magic 地址, 然后写死在核 0 的用户初始化代码上。

在完成所有上述操作后,core0 需要对每个核的 IPCGRx 寄存器写中断以唤醒其他核的 正常运行状态。IPCGRx 寄存器的 31-4 比特位是 IPC 中断源索引,最多可支持多达 28 个中 断源,文中例程可以设置为全0;比特3-1是保留位,可以任意赋值。因此只要对最低比特 赋1就可以完成IPC中断的触发。

140

145

150

155

160

165

170

31	30	29	28 2	7 8	3 7	6	5	4 :	3 1	1 0
SRC27	SRC27	SRC27	SRC27	SRC23-SRC4	SRC3	SRC2	SRC1	SRC0	RSV	IPCG
	图 4 DC 由躯开出家方思									

图 4 IPC 中断生成寄存器

到此为止基于 SPI 多核启动的过程全部结束,多核 DSP 正常运转。注意每个核的 IPC 中断生成寄存器有固定的内存映射地址,如下表所示:

表 2 IPCGR 奇仔希内仔映射							
起始地址	结束地址	长度	寄存器名	描述			
0x02620240	0x02620243	4B	IPCGR0	IPC核0中断生成寄存器			
0x02620244	0x02620247	4B	IPCGR1	IPC 核 1 中断生成寄存器			
0x02620248	0x0262024B	4B	IPCGR2	IPC核2中断生成寄存器			
0x0262024C	0x0262024F	4B	IPCGR3	IPC核3中断生成寄存器			
0x02620250	0x02620253	4B	IPCGR4	IPC核4中断生成寄存器			
0x02620254	0x02620257	4B	IPCGR5	IPC 核 5 中断生成寄存器			
0x02620258	0x0262025B	4B	IPCGR6	IPC核6中断生成寄存器			
0x0262025C	0x0262025F	4B	IPCGR7	IPC核7中断生成寄存器			

表 2 IPCGR 寄存器内存映射

# 2.2 管脚配置

SPI 启动的流程需要在两处修改管脚配置(EVM 板的拨码开关)。第一次是在烧写镜像工程进入 Flash 时,需要将 EVM 板调成 No-Boot 模式。承前所述,镜像工程必须实现预读到 DDR 里再烧写进 Nor flash,只有 No-Boot 模式才能初始化 DDR;第二次是成功烧写多核镜像后进行 reboot 的过程,将拨码开关配置成 SPI 模式才能将镜像成功读取并启动。

C6678 多核 DSP 的 ROM boot loader 驱动支持多种启动方式,参数的配置和读取都是由固化在 ROM 里的代码控制。Boot 模式信息来源于 BOOTMODE[12:0]寄存器,此寄存器的值包含 Boot 设备、设备配置、PLL 配置等参数。如下图所示:

	Boot 管脚											
12	11	10	9	8	7	6	5	4	3	2	1	0
PLL Mult I2C Ext Dev Cfg 设备配置								Boot 设备				

图 5 c6678 启动管脚配置

BOOTMODE[2:0]用于区分各种启动设备,如 SRIO、SPI、EMIF、PCIE 等。文中研究的 SPI 启动模式对应低 3 比特为"110"; BOOTMODE[9:3]用于配置设备,由于不同的设备具有不同的配置,因此这 7 比特的定义会因启动设备的不同而不同; 高 3 比特包含了一些锁相环总线等配置信息。BOOTMODE[12:3]可以描述为:

12	11	10	9	8	7	6	5	4	3
模式		4,5 pin	地址宽度	片	选		参数表	長索引	

图 6 SPI 启动管脚配置

模式选择涉及到 SPICLK 与上升沿的关系, SPI 通常选择为 4pin 模式, 地址线宽度为 24bit。由于只挂载一块 flash, 所以片选信号固定为 1。参数表的索引对应 boot 需要的参数 表,通常为零。根据拨码开关与 BOOTMODE 寄存器的关系,总结拨码开关处于 SPI boot 模式时的状态应该为 SW3~SW6: 1011 0000 0010 1000 (0 对应开 1 对应关)。

下图是 No-boot 模式下的管脚配置:

# 山国赵技论文在线

9	8	7	6	5	4	3
子	子模式			保	留	

图 7 No-boot 管脚配置

175

180

经过对照总结拨码开关处于 No boot 模式时的状态应该为:

SW3~SW6: 1000 0000 0000 1100 (0对应开 1对应关)。

### 2.3 参数表

下图的 Boot 参数表用于配置 SPI boot,由于 SPI boot 是寄存器的直接读写,因此配置过程中不会涉及到 EDMA 寄存器的配置。 Boot 参数表为 8 个字(32Bytes),位于烧写入flash 镜像的开头。不同的字段具有不同的意义:

字节偏移量	名称	描述
12	Options	选项
14	Mode	SPI 模式
16	Address Width	地址宽度
18	Data Width	数据宽度
20	NPin	管脚模式
22	Chipsel	片选信号
24	Read Addr MSW	读地址高位
26	Read Addr LSW	读地址低位
28	CPU Freq MHz	CPU 主频
30	Bus Freq, MHz	总线频率 M 单位
32	Bus Freq, kHz	总线频率 K 单位

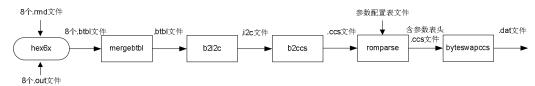
表 3 SPI boot 参数表

185 这些参数部分可以通过读取图 5 所示管脚配置来填充,也可以被用户自定义修改。而且 自定义修改可以覆盖拨码开关所表示的配置值。

注意无论是管脚配置还是参数表的书写都是为了生成 boot 参数表。Boot 参数表是由 32 行"字"表示,在实际实现的过程中可以定义一个解析工具,将参数配置简化。

### 2.4 工具链

190 由图 2 我们知道,工具链的将生成的文件转化为 Bootloader 可以"理解"的格式是 c66x 启动至关重要的一步。与以前的 DSP 启动相比,c66x 系列的工具链更加复杂和多样化。文件转换格式也呈现多态化。以.dat 文件格式为例,生成所需的镜像文件需要以下工具链做支持:



195

200

图 8 镜像工具链转化图

hex6x 文件需要和.rmd 文件配合使用,后者描述了 boot 参数表模式,ROM 宽度,大小端模式等信息。由此得到 8 个核的 btbl 文件包含了所有的内容信息,再经过两个小工具的转化可以得到有效数据信息。此时的.ccs 文件只包含各个段的内容,不包含 Boot 参数的任何内容,因此需要将一定格式的参数配置信息进行解析,作为 boot 参数表头加在.ccs 文件上成为一个含有参数配置头的.ccs 文件。最后由于 ROM bootloader 只识别大段模式数据,所

有需要进行一次大小端的转化。至此一个完整的可以被 ROM Bootloader 识别的多核镜像文件就成功生成。其格式如下图所示:

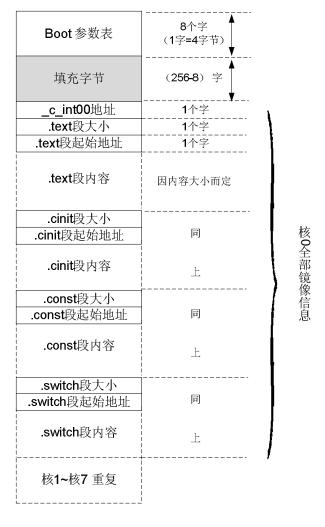


图 9 最终可烧写镜像格式图

# 3 工具链的优化和改进

在实际进行工具链转化的过程中发现,转化工具链部分工具大小受限。比如 b2i2c .exe 最大长度为 0x20000 个字节,如果只是单核镜像一般不会超出这个范围;但是八个核的镜像链接在一起就有可能超出范围,造成工具链转化文件出错。解决办法是找到 b2i2c 的源文件并修改大小上限,修改 MAX\_SIZE 宏以支持 8 核镜像的大小。

另外实际操作过程中,当固定 SPI 参数表的的时候,不用每次去解析参数配置文件。只需要生成一次 16 进制的参数表头,保存后加在经过大小端格式转换的数据文件之前也可以组装成一个完整的可烧写镜像文件。由此,原工具链可以优化并改进为:

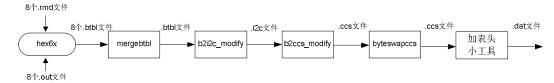


图 10 优化改进的工具链

205

210

该种工具链不受镜像大小约束并且减少了一步转化过程,变得更加方便快捷。

# 4 结论

220 本文给出了 TI 最新款 keystone 系列 c66x 多核 DSP 的 SPI Nor-flash 启动过程的研究以及工具链的使用和优化改进。该研究使得多核 DSP 复杂的启动过程变得清晰明了,简化了用户开发引导代码的难度;同时工具链的修改和优化使得多核启动能面向大容量镜像的烧写而不会有任何的限制。

## 225 [参考文献] (References)

- [1] Texas Instruments Incorporated TMS320C6000 系列 DSP 编程工具盒指南 田黎育 何佩琨 朱梦宇 北京:清华大学出版社,2006
- [2] 钟俊.TMS320C672x DSP 引导程序设计[D].合肥:中国科技大学,2010
- [3] Texas Instruments Incorporated, SPRUGY5A,keystone Architecture bootloader. Texas 230 Instruments,Dallas,Texas,2012.
  - [4] Texas Instruments Incorporated, SPRS691B,TMS320C6678 Data manual. Texas Instruments,Dallas,Texas,2012.
  - [5] Texas Instruments Incorporated, SPRU186P,TMS320C6000 Assembly Language Tools. Texas Instruments,Dallas,Texas,2006.

235