

一种基于 DSP 的全局动态可重构方法

符超*, 杨涛, 钱宏文, 吴翼虎, 付强

(中国电子科技集团公司 第五十八研究所, 无锡 214035)

摘要: 可重构计算技术很好地解决了计算性能、效能, 以及灵活性之间的矛盾, DSP 作为异构多核平台中极为重要的组成部分, 却少有面向其可重构的研究, 限制了该技术的应用。针对此问题提出一种基于 DSP 的全局动态可重构方法。预先分区 FLASH 并烧写 SPI 参数文件以及镜像文件 A、B, 系统启动后先运行镜像 A, 当收到重构命令时修改 SPI 参数, 将 BOOT 使用的镜像指向 B, 最后控制 DSP 重新 BOOT 以完成重构。使用 TI TMS320C6678 DSP 做测试, 结果表明, 该方法可以在 320 ms 内完成 DSP 内运行镜像的切换, 具有良好的应用前景。

关键词: DSP; 动态可重构; TMS320C6678

中图分类号: TP311.1

文献标识码: A

Global dynamic reconfigurable method based on DSP

FU Chao, YANG Tao, QIAN Hongwen, WU Yihu, FU Qiang

(No. 58 Research Institute, China Electronics Technology Group Corporation, Wuxi 214035, China)

Abstract: Reconfigurable computing technology solves the contradiction between computing performance, efficiency and flexibility very well. As a very important part of heterogeneous multi-core platform, DSP has few researches on its reconfiguration, which limits the application of this technology. To solve this problem, a global dynamic reconfiguration method based on DSP is proposed. Pre-partition FLASH memory and burn SPI parameter files as well as image files A and B. After the system starts, it runs image A first. When receiving the reconstruction command, modify SPI parameters to point the image in BOOT to B. Finally, control the DSP to restart BOOT to complete the reconstruction. The test results of TI TMS320C6678 DSP show that this method can complete the switching of DSP running image within 320 ms, and has a good application prospect.

Keywords: DSP; dynamic reconfiguration; TMS320C6678

0 引言

可重构计算是指能够实现算法到计算引擎的空间映射, 并在被制造成集成电路后还具备定制能力的计算组织形式。它可以在一定范围的应用集合中灵活切换, 实现计算资源的高利用率^[1]。动态可重构是可重构计算的一个分支, 其实现了硬件资源的时分复用, 能够有效提高嵌入式系统的灵活性和可靠性, 使系统在付出最小硬件代价的基础上实现最为高效的多任务处理能力^[2]。

动态可重构技术最早应用于 FPGA 平台, 已经有很多学者做了相关研究, 其研究成果广泛应用于航空航天、

电力电子工程、计算机科学等前沿科技领域^[3-7]。但是, 在常用的 DSP+FPGA 异构多核平台中, 由于业务是一个整体, 仅仅将 FPGA 程序重构并不能让整个业务按需切换, 因此研究面向 DSP 的动态重构方法迫在眉睫。

2015 年, 陆振林等提出了一种可重构的 DSP 任务动态加载方法^[8], 实际上是使用了 DSP 的 HPI (Host-Port Interface, 主机接口) BOOT, 而不是真正意义上的重构; 同年, 王国洗提出了一种基于 DSP 的动态重构框架^[9], 目的是通过在线传输动态库文件方式实现 DSP 的重构, 但是只提供了框架的设计以及动态库的解析、存储, 并没有完全实现重构功能; 2017 年, 白生炜提出了一种基于多核

符超(工程师), 主要研究方向为 DSP 软件开发、信号与信息处理; 杨涛(助理工程师), 主要研究方向为嵌入式软件开发; 钱宏文(研究员高级工程师), 主要研究方向为集成电路应用及微系统; 吴翼虎(高级工程师), 主要研究方向为嵌入式计算机和微系统; 付强(高级工程师), 主要研究方向为 FPGA 软件设计。

* 通信作者, E-mail: fuchao_x@foxmail.com。

DSP 的局部重构方法^[10],该方法以核 0 作为控制核,完成其余核程序的重构,但由于核 0 程序无法更改,因此应用范围有限;2018 年,何双宗、王巍等将参考文献[10]中重构方法应用到高速机床精准控制系统中^[11];2022 年,黄柏林提出了一种多核 DSP 的局部动态可重构技术^[12],但也只是将参考文献[10]中的内容应用到了信号调制识别领域,二者都没有提出新的 DSP 重构方法。

在以上研究基础上,本文提出了一种基于 DSP 的全局动态重构方法。通过合理分配 FLASH 内存分区存储 BOOT 参数数据、镜像数据,以 DSP 重新 BOOT 的方式,实时切换 DSP 中运行的应用程序。该方法可以很好地应用于异构多核架构下业务逻辑的动态重构。

1 可重构原理

全局动态可重构的目的是在 DSP 程序运行期间、在不断电的前提下,将芯片中正在执行的软件镜像(以下称执行镜像)完全替换为一个与其功能不同的镜像(以下称重构镜像),以实现业务逻辑的实时切换。在此,需要考虑的地方有两点:一是重构镜像文件如何获得;二是如何控制 DSP 实现镜像的替换,或者说,如何完成重构过程。

1.1 如何获得重构镜像

DSP 的软件镜像通常是使用官方提供的烧写工具链将应用工程编译生成的 .out 文件转换成 .dat 格式的二进制文件,其使用方法是先将镜像烧写固化至外部 FLASH 中,然后以 SPI(Serial Peripheral Interface,串行外设接口)加载、EMIF(External Memory Interface,外部存储器接口)加载等芯片规定的加载方式启动。如果在开始重构时才将重构镜像通过外设接口传入 DSP 内部,由于镜像数据的传输会占用一部分时间,势必会导致重构总时间增加,影响效率,因此考虑将重构镜像提前烧写至 FLASH 中,以提高重构过程的实时性。而执行镜像同样在 FLASH 中烧写固化,因此如果重构镜像还按照原方式烧写,会覆盖掉执行镜像导致下次上电后 DSP 无法正常启动。

为了保证重构镜像与执行镜像都正常使用,需要在 FLASH 中划分不同的区域来存储它们,例如执行镜像占用区域 0,重构镜像占用区域 1,二者在内存地址上严格隔离,互不重叠。这样系统上电后先运行执行镜像,在业务需要时再根据外部动作触发重构操作,转而执行重构镜像;而在重构镜像运行时,也可以按需再次重构至执行镜像。

1.2 重构过程

市面上用的比较多的 DSP 芯片,如国外 TI 公司的 C66x 核系列 DSP、国产国防科大 FT 系列 DSP,在程序运

行期间,镜像文件的各数据段(如 .text、.data、.stack 等)存储位置由程序 .cmd 文件 SECTIONS 部分中的内容来规定,具体地址可查看编译生成的 .map 文件。其搬移(即从 FLASH 复制到片上内存或者 DDR 中)依靠 DSP 片上固化的 RBL(ROM Bootloader)来完成。DSP 上电后,RBL 按照“段”的格式处理镜像文件数据:先扫描镜像文件得到段长度、段地址,再将相应长度的段数据搬移到 DSP 内部存储器对应段地址中。处理完成后程序指向入口地址 _c_int00 开始执行,DSP 即成功启动。

要动态完成镜像重构,可以按照以下思路进行:首先,添加一个外部触发动作来执行重构操作,比如中断或者自定协议命令;其次,人为实现 RBL 功能,编写对应的程序将重构镜像各数据段从 FLASH 中读出并搬移;最后,等待数据搬移完成再将程序跳转至重构镜像的入口地址 _c_int00 开始执行。这样做有两个难点:一是镜像文件需要按段准确解析,以获得每个段的信息;二是需要控制程序精准跳转到指定位置,但这并不好实现。

容易看出,动态完成镜像替换的过程相当于执行了一次 DSP 程序的 BOOT 流程。那么如果可以在动态重构时将 BOOT 选择的镜像从执行镜像改为重构镜像并控制 DSP 重新 BOOT,那么就可以利用 RBL 很方便地完成重构过程。

2 可重构实现

以重新 BOOT 的方式完成动态重构过程,从理论上讲切实可行,但不同的 DSP 芯片其支持的 BOOT 方式并不完全相同,因此需要选择一种确定的 BOOT 方式使用。

本文选择 TI TMS320C6678 芯片为硬件载体,以通用的 SPI BOOT 方式为例,详细介绍动态重构的实现方法。同系列的其他 DSP(如 C6657、C6655 等)启动方式通用且 DSP 核心同为 C66x 核,可复用此方法。

TMS320C6678 是 TI 公司生产的一款多 C66x 核 DSP 芯片,集成 8 个核心,主频最高可达 1.4 GHz,定点计算速度为 44.8 GMAC/核,浮点计算速度为 22.4 GFLOP/核,核内计算单元相互独立,各核独有 32 KB 的 L1D 以及 L1P 内存,512 KB 的 L2 SRAM,多核共享 4 MB 的 MSM SRAM 以及最高 8 GB 的 DDR 内存^[13]。

2.1 FLASH 内存分配

动态重构时要将执行镜像以及重构镜像都存储在外部 FLASH 中,二者的存储区域不能有重叠,否则会导致 DSP 程序无法正常启动。

DSP 在 SPI BOOT 时使用的镜像文件格式如图 1 所示,前 1 KB 空间存储的是 SPI 的 BOOT 参数,1 KB 以后

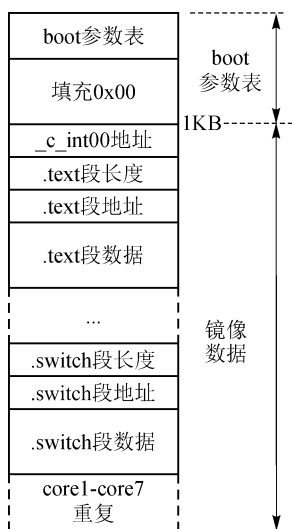


图 1 镜像格式图

Fig. 1 Mirror format chart

则是镜像的各段数据。

DSP 启动时, RBL 先固定从 0 地址的 BOOT 参数中读取 SPI 的各配置信息, 初始化好 SPI 接口, 然后按照镜像数据的结构按段搬移。因此, SPI 参数作为启动时必须使用到的数据无法存储至别的内存位置, 而镜像数据可以按需存储到任意位置。

通过改动 TI 官方的烧写工具链将每次转换时生成的烧写 .dat 文件分成两个独立的部分: SPI 参数文件 spi. dat 和生成的镜像数据文件 image. dat, 前者固化至 FLASH 的 0 地址, 后者则固化至规划的其他地址。由于各镜像文件的 spi. dat 完全相同, 因此可使用任意一份, 重构时只需使用重构镜像的 image. dat 即可。整个 FLASH 的内存分配如图 2 所示。

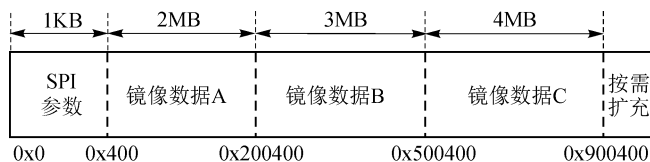


图 2 FLASH 内存分配图

Fig. 2 FLASH memory allocation diagram

图中, SPI 参数固定占用 FLASH 前 1 KB 区域, 镜像数据 A 至镜像数据 C (执行镜像或重构镜像) 占用的 FLASH 区域大小是根据其 .dat 文件设置的, 比如镜像文件 A 查看后为 1.1 MB, 于是可分配 2 MB 的空间供其使用。只要可以保证与镜像数据 B、C 等占用的 FLASH 区域不重叠, 大小可任意指定。

2.2 软件框架

动态可重构的重构过程是动态的, 即在系统软件执行过程中完成重构。这要求软件至少完成以下 3 个功能:

①保证 DSP 正常工作; ②与外部通信, 按照既定协议解析命令字, 并以一个特定的命令字来触发动态重构这个动作; ③控制 DSP 完成动态重构。

经过各方考量, 设计完成的 DSP 软件框架如图 3 所示。

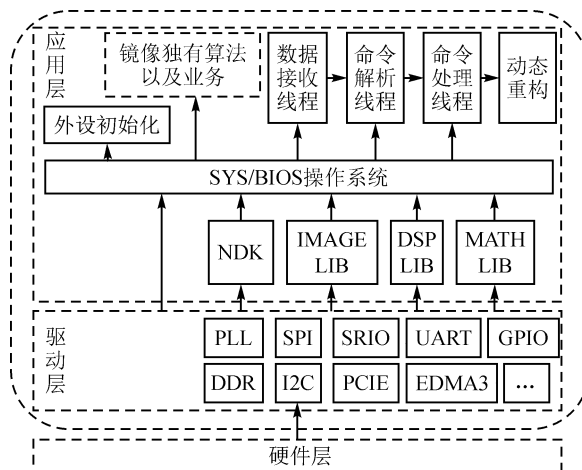


图 3 软件架构图

Fig. 3 Software architecture diagram

从图中可看出, 应用层与驱动层共同构成了系统的软件。执行镜像与重构镜像除了镜像独有算法、业务不同, 其余部分完全一致。

系统上电后, 先启动 SYS/BIOS 操作系统, 初始化各 DSP 外设, 再建立数据接收线程、命令解析线程以及命令处理线程, 最后将系统控制权交给操作系统, 由其完成各线程的调度。

软件运行时, 数据接收线程通过对外接口接收协议数据以及信号数据, 分别放进协议缓存与数据缓存; 命令解析线程从协议缓存中取数并解析命令字, 将有效命令字放进命令缓存; 命令处理线程从命令缓存中取出命令后执行对应操作, 比如此时取出的命令是重构, 则执行动态重构。3 个线程同时运行, 配合完成整个软件的功能。

在动态重构时, 先修改 FLASH 内 SPI 参数将 DSP 启动使用的镜像从执行镜像改为重构镜像, 再把执行镜像占用的 DSP 资源释放, 最后开始重新 BOOT。由此可得到动态重构的流程如图 4 所示。

3 测试与验证

3.1 硬件连接与测试方法

使用自研高速信号处理卡完成动态可重构的测试, 系统的硬件连接如图 5 所示。整个板卡由两片 TMS320C6678 并联组成, 上电时序由板载 CPLD (Complex Programming Logic Device, 复杂可编程逻辑器件) 控制, DSP 之间互连 HyperLink 高速口和若干 GPIO (Gener-

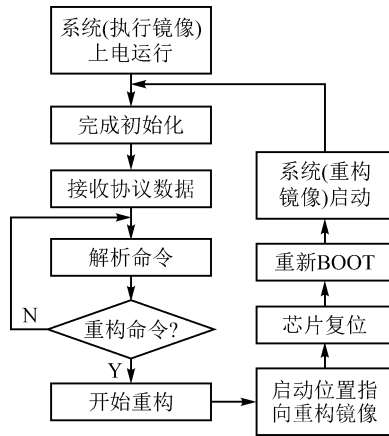


图 4 动态重构流程图

Fig. 4 Dynamic reconstruction flowchart

al – Purpose Input/Output, 通用 I/O 口)。此外,还设计了 SRIIO(Serial RapidIO, 串行 RapidIO)交换机以及网络交换机,以满足 DSP 与外部设备之间数据交互的需求。

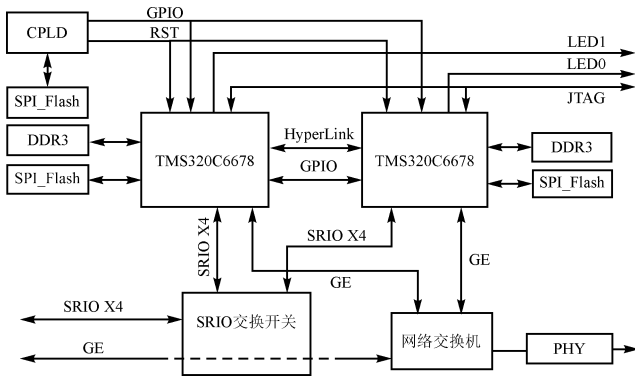


图 5 硬件连接框图

Fig. 5 Hardware connection block diagram

在实际测试时,提前将 A 镜像与 B 镜像烧写至 FLASH 中。DSP 通过千兆以太网与上位机实现协议数据的传输,其中 DSP 为 TCP 客户端,上位机为服务器。镜像 A、B 以前面板 LED 灯闪烁的频率来互相区分。为了准确呈现测试结果,除 LED 灯外,程序中还额外添加了向 MSM SRAM 中写 128 字节数据的操作,A 镜像从 0x0C18 0000 位置开始写入 0x0~0x7F 的递增数,B 镜像则在同样位置写入 0x7F~0x0 的递减数。同时通过 GPIO 引脚向板载 CPLD 发送开始重构以及结束重构信号,由于板卡不断电,因此 CPLD 可准确获取重构的起始时间点与结束时间点,相减即得到整个重构过程耗费的时间。

板卡上电后,DSP 先运行 A 镜像,然后通过上位机向 DSP 发送动态重构命令;DSP 接收到命令,按照协议解析到 B 镜像在 FLASH 中的存储位置,随即开始重构;重构完成后,DSP 内部运行的程序就从 A 镜像替换为了 B

镜像。

值得一提的是,在 B 镜像运行时依然可以通过上位机发送重构命令,控制 DSP 重新重构至 A 镜像。

3.2 测试结果

按上述步骤执行测试流程,测试结果如图 6、图 7 所示。

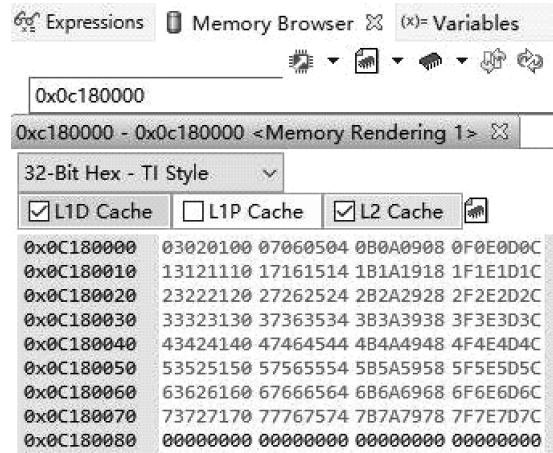


图 6 重构前内存图

Fig. 6 Memory map before refactoring

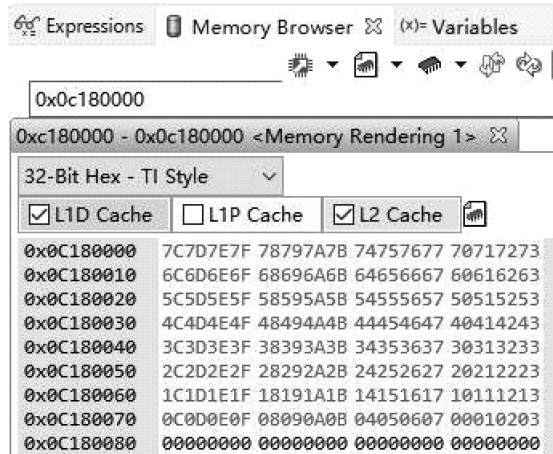


图 7 重构后内存图

Fig. 7 Reconstructed memory map

通过 CCS 的 Memory Browser 界面查看共享内存中的数据。由于 DSP 工作在小端,因此在 32 位模式显示时,每 4 个字节的数据显示顺序与内存中数据顺序刚好相反。如图 6 中 0x0C18 0000 位置,内存中数据为 0x0~0x3 四个 8 位数,显示为 32 位数 0x0302 0100。

由图中内容可以看到,重构前内存中写入的是递增数,而重构后写入的是递减数,与预期结果相符。也就是说,在重构前,DSP 中运行的是 A 镜像,而重构完成后,DSP 中的程序已完全切换为 B 镜像。

使用到的测试程序镜像文件大小约为 3.7 MB,经过

多次测试并统计动态重构耗费的时间,如表 1 所列。

表 1 重构耗时

Table 1 Reconstruction time consumption

统计次数/次	1	5	10
重构时间/(ms)	317	1 588	3 174

从收到重构命令后执行重构流程开始,到 DSP 程序重构结束,平均用时不到 320 ms。此时间长短与重构镜像大小及 SPI 启动时设置的 SPI 速率有关。从目前的测试结果来看,文中动态重构方法可满足大部分应用场景的实时性需求。

4 结 语

文章以 TMS320C6678 芯片为例,详细阐述了全局动态可重构方法的软件流程及实现,并做了测试。测试结果表明,文中所述方法可在 320 ms 内完成 DSP 业务逻辑的切换,但在一些需要局部算法重构的应用场合下,这种方式效率较低。如何有的放矢地重构 DSP 业务逻辑中的部分关键模块,将是下一步研究的内容。

参考文献

- [1] 高彦钊, 郭江兴, 刘勤让, 等. 计算机体系架构研究综述与思考[J]. 中国科学: 信息科学, 2022, 52(3): 377 - 398.
- [2] 赵谦, 邓豹, 刘婷婷. 嵌入式可重构信号处理计算机技术[J]. 计算机工程与设计, 2020, 41(11): 3289 - 3293(in Chinese).
- [3] PEZZAROSSA L, SCHOEBEHL M. A Controller for Dynamic Partial econfiguration in FPGA Based Real - Time Systems[C]//IEEE International Symposium on Real - time Distributed Computing. Toronto: IEEE, 2017: 88 - 96.
- [4] 李光明, 弓皓斌, 袁凯, 等. 基于 FPGA 的远程可重构无线传感器设计方法[J]. 无线电工程, 2023, 53(5): 1131 - 1137.
- [5] 虞业添, 施敏华, 郑倩云. FPGA 软件重构验证方法研究[J]. 电子设计工程, 2020, 28(21): 65 - 69.
- [6] 刘尧, 蒋林, 李远成, 等. 自适应二进制算术编码的动态可重构实现研究[J]. 电子测量技术, 2022, 45(19): 6.
- [7] 郝梦坛. 能量粒子探测器 FPGA 软件系统重构技术研究[D]. 北京: 中国科学院大学, 2022.
- [8] 陆振林, 赵元富, 兰利东等. 基于可重构技术的 DSP 任务动态加载方法研究[J]. 电子技术应用, 2015, 41(10): 24 - 26.
- [9] 王国洗. 嵌入式多核 DSP 动态重构框架的研究与实现[D]. 西安: 西安电子科技大学, 2015.
- [10] 白生炜. 多核 DSP 局部重构系统的设计与实现[D]. 天津: 天津工业大学, 2017.
- [11] 何双宗, 王巍, 田会娟, 等. 多核 DSP 的动态重构方案的设计与实现[J]. 机械设计, 2018, 35(S1): 29 - 32.
- [12] 黄柏林. 面向信号调制识别多核 DSP 处理的局部重构功能设计与实现[D]. 西安: 西安电子科技大学, 2022.
- [13] Texas Instruments. TMS320TMS320C6678 Multicore Fixed and Floating - Point Digital Signal Processor, 2012.

(责任编辑: 薛士然 收稿日期: 2023-09-14)